NBSIR 75-780

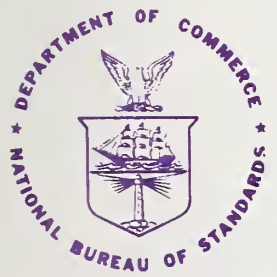
Mathematics and Engineering in Computer Science

Christopher J. Van Wyk

Institute for Computer Sciences and Technology
National Bureau of Standards
Washington, D. C. 20234

August, 1975

Final Report



U S. DEPARTMENT OF COMMERCE
NATIONAL BUREAU OF STANDARDS

NBSIR 75-780

MATHEMATICS AND ENGINEERING IN COMPUTER SCIENCE

Christopher J. Van Wyk

Institute for Computer Sciences and Technology
National Bureau of Standards
Washington, D. C. 20234

August, 1975

Final Report

U.S. DEPARTMENT OF COMMERCE, Elliot L. Richardson, *Secretary*
James A. Baker, III, *Under Secretary*
Dr. Betsy Ancker-Johnson, *Assistant Secretary for Science and Technology*
NATIONAL BUREAU OF STANDARDS, Ernest Ambler, *Acting Director*

I. Introduction

The objective of the research which culminated in this document was to identify and characterize areas of mathematics and engineering which are relevant to computer science. The descriptions of technical fields were to be comprehensible to laymen, yet accurate enough to satisfy technically knowledgeable personnel, and informative for all readers.

The division of the fields, shown in the table of contents, is somewhat arbitrary, but it is also quite convenient. A short explanation might be helpful.

The mathematical fields can be grouped by noting the importance of a field to computer science, and, to a lesser extent, the existence of the field independent of computer science. For example, Boolean algebra and information theory could (and did) exist without computers being available. On the other hand, most problems in artificial intelligence or optimization, once they reach even moderate complexity, must be solved with a computer, since hand solution would take too much time.

The criteria for grouping the engineering fields are similar. Those which have "lives of their own," like circuit theory and signal processing, are classified as contributing to computer science, whereas engineering fields like microprogramming and software engineering which are dependent for their existence on computers are listed as components of computer engineering.

It is important to note that many of the fields covered herein do not have a generally accepted definition. Hence, the definitions stated

have been culled from the listed references, and joined into a coherent statement which hopefully delimits the field neither too broadly nor too narrowly. In cases where serious differences about a field's subject matter were found in the literature, a footnote indicating the source of the various components of the definitions has been written.

In a work such as this, no list of references could be complete; hence, no attempt has been made to include all relevant references. Instead, works have been cited because they are classics in the field or they provide a good introduction to it. Most of the references listed in the bibliography are books; many contain comprehensive bibliographies which reference journal articles. Most journal articles were not included because they dealt with small areas of a field, rather than providing a comprehensive view, like books.

This work could not have been completed in a summer without the help of many members of the Institute for Computer Sciences and Technology. Among those I would like to thank are: Dr. Ruth M. Davis, for giving me the opportunity to do this work, and supporting me throughout the effort; Dr. Joseph O. Harrison, Jr., who spent many hours reviewing drafts of each section, suggesting (though never demanding) changes and clarifications which needed to be made; Dr. Marshall Abrams, Dr. James Albus, Dr. John Evans, Mr. S. Jeffery, Mr. George Lindamood, Dr. Gordon Lyon, Mr. Paul Meissner, Mr. Thomas N. Pyke, Jr., and Dr. Selden Stewart, who helped me with particularly difficult topics, reviewed sections related to their areas of expertise and suggested improvements; Mmes. Jo Ann Brooks, Elaine Frye, Beverly Geisbert, Shirley Radack, and Anita Walker, who typed the many

drafts and the final copy of this report; and the staffs of the Bureau and Institute libraries, as well as many individuals, who allowed me to search through their collections for reference materials.

Table of Contents

II. How To Use This Document

There exists today much ambiguity about the province of computer science and its relationships with mathematics and engineering. This document attempts to identify areas of mathematics and engineering which are related to, overlap with, or are even considered to be part of computer science. An overview of the areas so identified is given as a guide to the professional who needs to know about computer science or some aspect of it, either for himself or to explain it to others; the references suggested direct the reader to more detailed explanations of the area.

There are fifteen mathematical areas treated in Section IV: twelve as contributing to computer science (IV.A),and three as dependent on it (IV.B). Similarly, there are nine areas of engineering treated in Section V, with four contributing fields (V.A) and five components of computer engineering (V.B). These areas are closely related; hence, while no attempt at explicit cross-references has been made, mention is freely made of the connections one area has with others. In this regard, a list of topics not treated separately (Section III) has been prepared; this list shows where descriptions of areas which do not have a complete write-up devoted to them, but which are nevertheless relevant to computer science, may be found.

Each description is in a standard format which has five components; this format was designed to allow ease of reference to specific information which a reader may seek. The five components are:

- general definition. As explained in the Introduction (Section I), this section attempts to given an idea of the subject matter of the area and the problems with which it is concerned.
- historical perspective. This section mentions prominent workers and significant accomplishments in the area. It also provides an idea of when the area became important or was recognized as discipline.
- principles and theories. This section gives a general look at significant or fundamental ideas and achievements in the area.
- applications to computer science. This section shows the importance of an area to computer science, and the relationship of the area to others in computer science.
- references. The abbreviated references are written out fully in the bibliography (Section VI). These references should prove a good source of additional information to the interested reader.

Draft copies of this document have already proven useful to professionals in the field of computer science. It is hoped that the scope of users may be broadened by issuance of the document in this form.

III. List of Topics Not Treated Separately

The list at left below contains the names of topics important to computer science but treated as subtopics of a field in this report. The section (or sections) wherein information about the topic may be found is indicated in the column at right.

To Read About:	See:	
Algebra	I.A.6.	Discrete Mathematics
Algebraic Linguistics	I.A.7.	Formal Language Theory
Binary Arithmetic	I.A.4.	Computer Arithmetic
Calculus	I.B.2.	Optimization Theory
Calculus of Variations	I.B.2.	Optimization Theory
Coding Theory	I.A.8.	Information Theory
Combinatorics	I.A.6.	Discrete Mathematics
Communication Theory	I.A.8.	Information Theory
Computational Linguistics	I.A.7.	Formal Language Theory
Data Structures	I.A.6.	Discrete Mathematics
Differential Calculus	I.B.2.	Optimization Theory
Distribution Theory	I.B.3.	Probability and Statistics
Dynamic Programming	I.B.2.	Optimization Theory
Filter Design	II.A.1.	Circuit Theory
Finite Mathematics	I.A.6.	Discrete Mathematics
Game Theory	I.B.2.	Optimization Theory
Geometry	II.A.3.	Display Systems Engineering
Graph Theory	I.A.6.	Discrete Mathematics
Heuristic Methods	I.B.1.	Artificial Intelligence
Holography	II.B.3.	Optical Computing
Linear Programming	I.B.2.	Optimization Theory

Logic	I.A.1. Theory of Algorithms I.A.3. Boolean Algebra I.A.7. Formal Language Theory
Markov Processes	I.B.3. Probability and Statistics
Modelling	I.A.12. Systems Theory
Multivariate Calculus	I.B.2. Optimization Theory
Network Theory	II.A.1. Circuit Theory
Non-Linear Programming	I.B.2. Optimization Theory
Pattern Recognition	I.B.1. Artificial Intelligence
Predicate Calculus	I.A.7. Formal Language Theory
Reliability Theory	I.A.5. Control Theory
Recursive Function Theory	I.A.2. Automata Theory
Residue Arithmetic	I.A.4. Computer Arithmetic
Set Theory	I.A.6. Discrete Mathematics
Simulation	I.A.12. Systems Theory
Stability Theory	I.A.5. Control Theory
Stochastic Processes	I.B.3. Probability and Statistics

IV.A.1. THEORY OF ALGORITHMS

General Definition

An algorithm is an explicit statement of the steps needed to accomplish a task. The theory of algorithms is concerned with the study of algorithms in terms of their efficient use of time and storage, as well as the proof that a given algorithm or its computer implementation works as intended. The design and improvement of algorithms is also an area of study in the theory of algorithms.

Historical Perspective

Recognition of the theory of algorithms as a field of study coincides with the advent of computers, since the problems posed for computer solution are much larger than those previously solved by hand, and because of the expense of computer time and storage.

Informal notions about algorithms and programs have existed since the days of the earliest computers, but Knuth and Floyd were among the first to suggest the desirability of a formalism for studying the properties of algorithms. Other contributors to the theory of algorithms include Aho, Dijkstra, Hoare, Hopcroft, and Ullman.

Principles and Theories

Two very important fields ancillary to the theory of algorithms are automata theory and formal language theory. Automata theory is the study of machines as information processing devices; many results have been established in it about minimal machine representations and about the unsolvability of certain problems. Formal language theory studies the structure of languages; among the languages studied (important to the theory of algorithms) is the first order predicate calculus.

In treatises on the theory of algorithms, an important first step is the design of a formal language for expressing algorithms. Different authors use different languages, but features shared by all include simplicity and completeness; simplicity makes existing algorithms more easily analyzable, while completeness assures that any algorithm is expressible in the language (Böhm and Jacopini established a fundamental result in the completeness of formal languages.).

The theory of algorithms studies both programming and data structuring techniques. Among the former are such concepts as recursion, iteration, and the evaluation of conditional expressions; lists, queues, and stacks are commonly used data structuring techniques. For example, in designing an algorithm to sort a list, a decision whether to sort the whole list at once, or to break it into smaller sublists and then to merge them into a final sorted list, would be based on the size of the list to be sorted. If the latter option were chosen, it could be accomplished either by recursion or by careful "bookkeeping." Recursion involves having a program call itself during execution; in the example, a recursive program to sort list L might invoke itself to sort the first half of list L; this invocation might in turn cause another call to sort the first quarter of list L; this technique has been called "divide and conquer" by Aho. "Bookkeeping" would involve keeping track of how far the list had been sorted - how close it was to the correct order - as the program progressed. Whereas recursion is often easier to program than the "bookkeeping," the efficiency of the program would depend on how the computer handles recursion. Such an abundance of options and important considerations is common in computer solutions.

Among the important procedures for study in the theory of algorithms are those for sorting, set manipulation, matrix operations, Fourier transforms, and arithmetic. A proof that an algorithm will perform the intended procedure is an integral part of the analysis, as are predictions about expected performance and performance in the worst possible case. (Often the algorithm with the best worst case performance will not have the best average performance in a collection of algorithms for performing a procedure.) Formal proofs of the correctness of an algorithm or program usually involve formal language theory, particularly the first-order predicate calculus.

Applications to Computer Science

The theory of algorithms is vital to computer science; in fact, Knuth claims that the two are identical. It is important in the design of software, particularly since many computer applications would be economically unfeasible or impossible, because of limitations on computer time and space.

References

Aho 1974
Dahl 1972
Knuth 1968
Knuth 1969
Knuth 1973
Miller 1972
Minsky 1967
Traub 1973

IV.A.2. AUTOMATA THEORY

General Definition

Automata theory is the study of machines (automata) and humans as information processing devices. Deterministic automata theory is concerned

with ideal machines: all input channels are controlled only by the user, and components always work as designed. Probabilistic automata theory, on the other hand, studies the more realistic case: input channels may be noisy, or components may fail unpredictably.

Historical Perspective

Though speculation about the workings of the human mind (analytic automata theory) is not new, it was only a decade before the advent of the electronic computer and attempts at modeling or imitation of thought processes (synthetic automata theory) that an abstract automata theory arose. Turing was the first to state a model of computation procedures, and all systems since proposed as models of such procedures have been demonstrated to be equivalent to his. These machine representation methods form the basis for automata theoretical studies today.

Principles and Theories

The notion of an effective computation procedure is central to automata theory. Intuitively, it may be thought of as a guaranteed way of solving a problem; for example, Euclid's algorithm to find the greatest common divisor of two numbers is an effective procedure. Turing's thesis is that effective procedures are those performed by Turing machines. Among the many formulations of Turing machines, a relatively common one is a device which has a finite number of internal states, a head which can read or write one symbol at a time on a tape which can be made infinitely long, and a specification telling what state the machine enters when it is in a given state and it reads a given symbol. The

machine is designed to operate so that input to the procedure is on the tape at the start, and output is on the tape after computation.

Many results concerning Turing machines have been discovered. Turing himself demonstrated the existence of a universal Turing machine--one which would simulate any Turing machine, given its specifications. Shannon proved that two machine states or a two element tape alphabet are sufficient for the construction of any Turing machine. A proof that no Turing machine exists which can solve the halting problem--that is decide whether a given Turing machine will halt on given input--was accomplished using methods analogous to those in Gödel's proof of the incompleteness of arithmetic.

Other methods of representing machines are due to Post, Kleene, and Smullyan; all have been shown equivalent to Turing's representation. The McCulloch-Pitts logical calculus can also be used to study machines, but was designed originally as a simplified model of human brain neurons. The McCarthy formalism is a convenient notation for indicating that certain relationships hold among numbers. Recursive function theory also provides a powerful tool for studying automata. Von Neumann was interested in the design of automata which could construct other automata or which could reproduce themselves.

In the area of probabilistic automata, an important result is Shannon's fundamental theorem for a discrete noisy channel (1948). The effect of unreliable components on automata was also of interest to von Neumann, who developed a method (multiplexing) of synthesising reliable automata from unreliable components. Cowan and Winograd also presented a method for reliable communication and computation on noisy channels.

Applications to Computer Science

Automata theory has applications in the design of reliable computer circuits, in algorithmic analysis, and in artificial intelligerce studies.

References

Arbib 1964
Davis 1958
Eilenberg 1973
Feigenbaum 1963
Minsky 1967
Nivat 1973
Shannon 1956
von Neumann 1956

IV.A.3. BOOLEAN ALGEBRA

General Definition

A Boolean algebra may be considered mathematically in many ways: for example, as a ring of idempotent elements, or as a complemented distributive lattice. Most often, however, its usefulness as a subclass of the propositional calculus of formal logic is exploited. (It is a subclass because it does not include, for example, syllogistic logic.)

Historical Perspective

Boolean algebra was first presented by Boole in 1847; it was a pioneering achievement as an algebraic calculus with possible non-arithmetical interpretations. It was improved by Jevons and Peirce, and later by Schröder.

Principles and Theories

Like all algebras, Boolean algebra operates on a set; in its logical context, this set consists of statements and is termed the universe of discourse. A fundamental assumption is that the universe of discourse

is the union of two disjoint classes: true statements and false statements.

A common use of Boolean algebra is the demonstration of the validity of an argument form; that is, given a set of true premisses, related to one another in the same way as the abstract form, what conclusions can validly be deduced? In making such determinations, it is convenient to represent statements by letters and relations among statements by connective symbols. Common connectives and symbols include:

not	$\sim$ ¬ — ′
or	$\vee$
and	$\cdot$ & $\wedge$
if . . . then	$\supset$ $\rightarrow$
if and only if	$\equiv$ $\sim$ $\leftrightarrow$ iff

An important property of logical disjunction is its inclusiveness: "p v q" means that p is true or q is true or both are true. This modification to Boole's original work was suggested by Jevons.

Certain axioms are used to transform propositions; these rules are true regardless of what interpretation is given to the statement variables; the simplest expresses the fundamental assumption above: $p \vee \sim p$. Two of the most important axioms are credited to DeMorgan: $\sim(p \,\&\, q) \equiv (\sim p \vee \sim q)$ and $\sim(p \vee q) \equiv (\sim p \,\&\, \sim q)$. The negation of an axiom is a contradiction--false no matter what interpretation is given the statement variables; e. g., $p \,\&\, \sim p$. The idea that a contradiction implies any statement was Peirce's. Schröder gave Boolean algebra the form of a deductive system.

The limitations of Boolean algebra may be seen in this example. The argument on the left below is valid in Boolean algebra; the one on the right is not.

If Aristotle is a man, then he is mortal.	All men are mortal.
Aristotle is a man.	Aristotle is a man.
Hence, Aristotle is mortal.	Hence, Aristotle is mortal.

The left hand argument is valid because it has the abstract form $p \supset q$; p; $\therefore$ q. The inference is allowed by a rule called modus ponens. But the right hand argument has the abstract form p; q; $\therefore$ r, which is not generally valid; Boolean algebra does not allow the implicit deduction that if Aristotle is a man then he is in the class of all men. Syllogistic logic, however, does allow this inference.

Applications to Computer Science

The main application of Boolean algebra in computer science is to switching theory. (Shannon was the first to suggest this.) A switching circuit produces output voltage depending on its configuration and the orientation of input voltages. For example, an AND-gate would not be on unless all inputs were on; an OR-gate would be on unless no inputs were on. Computers use switching circuits in the performance of arithmetic and in logical tests.

References

Boole 1847
Flegg 1964
Hammar 1968
Hohn 1966
Klír 1966
Schröder 1966

IV.A.4. COMPUTER ARITHMETIC

General Definition

Computer arithmetic is the study of number systems with a view toward their computer implementation for performance of basic arithmetical operations.

Historical Perspective

Many of the techniques of computer arithmetic have existed for centuries, being used to simplify or check hand calculations. The techniques are usually based on number theory.

Principles and Theories

The need for computer arithmetic is based on a fundamental difference between humans and computers: in the performance of calculations, a human can decide, at any point in a calculation, how many digits he wishes to carry (as significant) in computation; a computer must be instructed to carry a fixed number of digits, and usually must use the same number all the way through a calculation. Since the expense of computers is determined in part by the number of digits carried in calculations, a trade-off must be made between the accuracy of computations and their cost.

The binary number system is often used for computer arithmetic. The binary system uses two as its base, and is positional like the decimal system: each digit position corresponds to a power of two. Binary arithmetic is simple because the only digits used are zero and one; thus, addition and multiplication tables are small. Subtraction is also simple, because adding the two's complement of the subtrahend to the minuend yields the difference, and two's complementation is easy to perform.

In computer implementation, provision is often made for the representation of two types of numbers--fixed point and floating point. Fixed point numbers are usually considered to have the decimal point at the extreme right, thus representing integers. Floating point numbers are usually represented as the product of a proper binary fraction and a power of two; this allows a greater range of numbers to be represented and increases the available precision. Methods have been developed for scaling numbers represented in this manner, and for detecting overflow conditions which may occur as a result of performing arithmetic operations.

Residue arithmetic operates on number systems which are not fixed-base weighted like the decimal system; rather, the base is a vector of integers, and the residue representation of a number is a vector of remainders corresponding to division of the number by each element of the base. For example, in base (2,3), the decimal number 11 has the residue representation (1,2). Though each decimal number has only one residue representation, one residue representation can refer to many decimal numbers (59 is also represented by (1,2) in (2,3)).

A special case of residue systems are modular systems. In a modular system, the base is a single number. A familiar example of a modular system is a clock; for example, 8 hours past 7 o'clock equals 3'o'clock.

Obviously, the base for a computer system must be larger to allow the unique representation of many numbers.

Another aspect of computer arithmetic which has been extensively studied is conversion methods between binary and residue representations and the decimal system, important since computer output must be almost always be in decimal notation to be useful to humans.

Applications to Computer Science

Computer arithmetic is important in the development of computer hardware for performance of routine arithmetical tasks. The efficient design of arithmetic logic units has importance in the speed with which a computer will operate.

References

Borevich 1966
Chu 1962
Foster 1970
Kostopoulos 1975
Marcus 1973
Szabó 1967
van der Waerden 1970
Weil 1973

IV.A.5. CONTROL THEORY

General Definition

Control theory is an important component of systems theory, concerned with controlling physical systems' performance. Optimization methods play an important role in control theory. Reliability theory and stability

theory are subdisciplines of control theory.

Historical Perspective

Control theory was recognized as a discipline only in the 1950's; by then, system design had become so complex and costly that a systematic method of system design and control was needed to replace the heuristic methods then in common use.

Principles and Theories

In control theoretic analyses, the first step is the development of a mathematical model of the system to be studied; for this task, techniques from general systems theory are used to define the system and distinguish it from its environment, and to model the system using divers mathematical methods, including calculus, probability and statistics, and queueing theory.

The control theorist usually attempts to optimize the modelled system. Many of the techniques for optimization are borrowed from such areas as calculus, linear programming, and game theory. One of the most important control mechanisms available for optimization is feedback, which involves the use of part of a system's output as its input. For example, in the heating system for a house, the heat output by the system is detected by a thermostat which then triggers the furnace to produce more or less heat, depending on the temperature indicated as desirable.

Reliability theory is concerned with designing systems to meet high performance criteria. That is, their failure rate must be very low. Reliability theory deals with the effects of unreliable components in systems design, and with possible remedies, e.g., backup systems or multiplexing. Very often, tradeoffs between the reliability and the cost or speed of a system must be considered in control theoretic analyses.

Stability theory is concerned with the effects of disturbances on the systems equilibrium. A typical question might be: what happens when the system operates in conditions which are not those for which it was designed? Feedback is one of the important considerations in stability theory: one can imagine a feedback system which overcorrects in response to a stimulus, oscillates wildly instead of converging to the desired point and never accomplishes the task intended.

Applications to Computer Science

Control theory is used in computer science in the design of executive routines for computer systems, particularly in multiprocessing environments. It is also important in the design of input/output devices and in memory management systems using paging.

References

Barlow 1965
Bazovsky 1961
Bellman 1967
Chorafas 1966
Gumowski 1968
Kirk 1970

Lefschetz 1965
Lloyd 1962
Peschon 1965
Skinners 1972
Widnall 1968
Wiener 1948
Willens 1970

IV.A.6. DISCRETE MATHEMATICS

General Definition

Discrete mathematics is a modern branch of mathematics which theory is important in logic, set theory, algebra, graph theory, combinatorics,[1] probability, linear programming,[2] and formal language theory. In a sense, it may be regarded as forming the basis for all of those fields, as a theory which deals with discrete structures.

Historical Perspective

The recognition of discrete mathematics as a separate field coincides roughly with the rise of such disciplines as operations research and decision theory, in which mathematical techniques are applied to systems in discrete stages of development. Finite mathematics is a subfield of discrete mathematics which concerns finite systems. An example of such a system is a card game like poker, which consists of a finite card deck in discrete stages: shuffle, cut, deal, etc.

---

[1] Discrete mathematics and combinatorics are considered identical by Berman and Fryer (Berman 1972).

[2] Linear programming is mentioned as an element of discrete mathematics by Kemeny, Snell and Thompson (Kemeny 1966).

Principles and Theories

The logical theory underlying discrete mathematics is mainly Boolean algebra, i.e., a subset of the propositional calculus which operates on statements with regard only to their truth value, and not with regard to relationships between nouns within them. Quine and McCluskey contributed a method for minimizing Boolean functions: representing them using as few symbols as possible. Set theory is basically an application of Boolean algebra, enriched by syllogistic logic.

Algebra is a study of systems in terms of their structures--determining what is and what is not true about them. Usually once a few properties are established, others may be deduced to hold by analogy to other systems with similar properties. One example of an application of algebra in discrete mathematics is algebraic linguistics--the use of algebraic techniques on natural and formal languages. (English is a natural language; ALGOL a formal language.)

Graph theory deals with points (vertices) and connections between them (edges). As in algebra, certain characteristics of a graph determine the possibility of, for example, connecting any two points of the graph along a path consisting of existing edges. A particularly important area of graph theory in discrete mathematics is tree theory, which is concerned with certain types of graphs. Gorn is responsible for a redefinition of the tree concept.

Combinatorics is concerned with problems in the arrangement and selection of objects, as well as with their permutation and combination. As such, it may be regarded as providing the basis for probabilistic and

statistical studies. However, it is also involved in geometric problems like map coloring and with connections between sequence representations and recurrence relations, particularly the Fibonacci sequence. Winograd and Spira, and Minsky and Papert developed theories about finite network complexity.

Probability is an attempt to predict the likelihood of a particular event given knowledge of conditions leading to it. It is useful in designing mathematical models and in the computer simulation of physical systems. In operations research, linear programming and game theory are used in attempts to optimize situations.

The study of properties of data structures is based on many of the topics mentioned above. Two important data structures are queues and stacks. Queues usually accept data at one end and emit it from the other; they are sometimes said to have a first in-first discipline. Stacks generally accept and emit data from the same end; their discipline is referred to as last in-first out. Queues and stacks are special cases of lists, and another important topic is the determination of methods for representing lists in computers; this is particularly important when contiguous storage is not available for a list and pointers must therefore be established to indicate the location of list elements. Another application of the theory is to representing so-called sparse matrices (i.e., those with only a few non-zero elements with respect to their size) efficiently.

Applications to Computer Science

Discrete mathematics is relevant to computer science from an important design standpoint: any computer is finite and must perform operations sequentially. Other computer science areas where discrete mathematics is relevant include machine representation of data structures. Specialized discrete mathematical techniques have computer science applications in the design of switching circuits, automata theory, compiler construction, information transmission, and computer simulation.

References

Anton 1974
Berman 1972
Bertziss 1975
Bobrow 1974
Kemeny 1966
Korfhage 1974
Patton 1974

IV.A.7. FORMAL LANGUAGE THEORY

General Definition

Formal language theory attempts to discover and characterize the properties of languages (including formal, as well as natural, languages) in a mathematical way.

Historical Perspective

Formal language theory is closely related to automata theory; in fact, many of the major developments in formal language theory involve the demonstration that a class of languages is equivalent to a class of automata.

Formal language theory is also important in linguistics, for it has shown that certain problems in machine translation of natural languages cannot have algorithmic solutions. Two famous linguists who have also contributed to formal language theory are Chomsky and Bar-Hillel.

Principles and Theories

Formal language theory defines a language as a set of (usually finite) strings composed of elements from a finite alphabet. Two important methods of specifying a language are by means of generative grammars and recognition automata. A generative grammar is a set of rules (or rewriting system) which, when applied to an alphabet, generates all strings of the language. These rules may be viewed as the inverse of sentence diagramming: instead of separating a particular sentence into subject and predicate, a rule will state that a sentence in the language may be composed of subject and predicate; other rules then specify how to form subjects and predicates. Often, the set of all languages defined by rewriting systems is too large for convenient study, so restrictions are applied on the types of rules to be allowed in the grammar; the Chomsky hierarchy is an often used sequence for restricting languages.

Recognition automata are machines with a finite number of states, which accept input and produce output, and which may have extra storage. As with grammars, the limitation of what types of storage or input arrangements are allowed affects the variety of languages which can be recognized by an automaton; there exist machines which are equivalent to

each class in the Chomsky hierarchy. One of the most interesting problems in formal language theory is the determination of whether a language is recursive--whether there exists an algorithm for determining whether a given string is a language element--and if it is, whether an efficient procedure for recognizing all strings in the language exists.

In natural languages it is intuitively plausible[1] that one can determine whether a string is part of the language or not. Thus, attempts to design generative grammars for natural languages are an important area of study in linguistics; Chomsky's transformational grammar approach to language is one example of a generative grammar.

Some aspects of programming languages may be viewed in terms of formal languages, and developing generative grammatical representations for them is important in studying their properties; one such representation, appropriate only for certain types of languages, is due to Backus and Naur. LISP is a programming language which design is closely related to formal language theoretic considerations, particularly Church's lambda-notation.

Formal language theory is closely related to mathematical logic; in fact, formal language theory can be used in the design of systems for representing logical statements or arguments. Some formulations of the predicate calculus[2],

---

[1] According to Chomsky.

[2] A system of logic, deeply explored in Principia Mathematica by Russell and Whitehead, which includes Boolean algebra and Aristotelian syllogistic logic, but is richer because it allows the quantification of variables; for example, the sentence "for all real numbers x, there exists an integer i such that i is the greatest integer less than x" (basically, this guarantees that if a number is separated into whole number and proper fractional part, the whole number is unique) has a one-line predicate calculus representation, but no finite Boolean or Aristotelian representation.

as well as of Church's lambda-calculus, result from a language-theoretic point of view; the decidability or completeness of a language can then be analyzed in light of its relation to logical systems with known properties of decidability and completeness.

Applications to Computer Science

Formal language theory is useful in the design of computer programming languages and in the construction of parsers for their compilers. It also has applications in attempts to translate natural languages by machine and in efforts aimed at computer programs to comprehend natural languages. The use of linguistic techniques in data retrieval systems has also been reported.[1]

References

Bar-Hillel 1964
Book 1973
Chomsky 1965
Eilenberg 1974
Hopcroft 1969
Kurki-Suonio 1971
Lee 1972
Marcus 1967
NYU 1970
Nivat 1973
Ollongren 1974
Salomaa 1973

[1] See NYU 1970.

IV.A.8. INFORMATION THEORY

General Definition

Information theory is the study of properties of the symbols and media used in the storage and transmission of messages. Communication theory is a major part of information theory.

Historical Perspective

The equivalence of information and negative entropy was suggested by Szilard in 1929. The idea was rediscovered by Shannon in 1948, and he called it The Mathematical Theory of Communication. Wiener published another approach to information theory, but Shannon's is the generally accepted basis for a theory of information.

Principles and Theories

In information theory, a message is considered without regard to its possible correspondence to the real world; for example, the transmission and receipt of a one-bit is not interpreted as "yes" or "no" in answer to a question but is rather seen simply as a one-bit. The amount of information carried in a message is defined, statistically, as the negative logarithm of the probability of the transmission of the message; thus, an expected message is less informative than an unexpected one. The equivalence between information and (thermodynamical) negative entropy arises because entropy is a tendency toward disorder or randomness, and information is regarded as combatting this tendency--as increasing order in the world.

Communication theory deals with problems in message transmission. There are two important subtopics. One deals with the characteristics of the media used to transmit messages--channels. Among the features of a channel which are of interest are the capacity of a channel for information, the amount and type of noise present in a channel, the provision for memory (if any), and whether the channel accepts discrete or continuous transmission. Shannon's Fundamental Theorem states that arbitrarily high reliability may be achieved without degrading the transmission rate below a parameter called the channel capacity.

The other important area of communication theory deals with coding. Certain properties of codes, such as redundance and parity-checking, contribute to the reliability of message transmission. However, they also reduce the possible transmission rate. Coding design involves attempting to optimize the trade off between reliability and rate of transmission. Major results in coding include an extended Carnot principle: a gain in information is always accompanied by a larger gain in entropy; and the definition of Hamming distance: a measure of similarity between code elements.

A third concern of information theory is the analysis of information sources. An important property of a source is whether successive messages are independent. Another consideration is the number of possible messages versus the number of messages considered meaningful. Knowledge of such source properties is important in the design of optimal communication networks.

Applications to Computer Science

Information theoretic considerations are important to computer science in the design of codes and channels for transmission of data from one part of a computer to another and between computer and user.

References

Abramson 1963
Ash 1967
Brillouin 1962
Feinstein 1958
Fuchs 1971
Slepian 1973

IV.A.9. NUMERICAL ANALYSIS

General Definition

Numerical analysis is concerned with the (often approximate) numerical solution of equations; here, such problems as finding function values and inverting matrices are considered to be solving equations. Estimates of the error involved in the methods used are also of interest.

Historical Perspective

Before the digital computer, many numerical analytic methods had been developed for hand computation, and analyses of errors to which they were susceptible had been accomplished. With the advent of the computer, the adaptation of such methods to performance by computer was begun. This sometimes entailed the development of algorithms more suited to computer operation, and analyses of errors introduced by the finite nature of computer arithmetic.

Principles and Theories

Approximations to the values of common functions--trigonometric, logarithmic, etc.--are a primary concern of numerical analysis. Classical methods for representing functions include Fourier and power series, continued fractions, and rational functions; all of these consist of an infinite number of terms. It is the task of numerical analysis to determine how many terms are necessary for adequate precision or number of significant digits; generally, different approximations are appropriate for different regions in the functions' domain. For example, a series giving the sine of a small angle to six decimal places in ten terms might need hundreds of terms to achieve the same precision for large angles. Many polynomial approximations to functions are due to Chebyshev, Gram, Hermite, Laguerre, and Legendre.

Integration is a versatile technique which is also the subject of numerical analysis. Integration can be used in the determination of areas and volumes, and in the summation of series. Numerical approximations to integrals are often necessary for complex problems. A popular method for this is quadrature, for which methods have been developed by Gauss, Hermite, Jacobi, and Simpson. A related problem--the solution of differential equations--is found in almost all branches of engineering, chemistry, and physics. Runge-Kutta methods are a popular solution method based on Taylor series expansions. Predictor-corrector methods, as the name implies, give good approximations by predicting a value, then correcting it by estimating the error in it.

Interpolation is a technique used to obtain the approximate value of a function between known values. For example, one might want to find the sine of 30°30' and only have values of the sine at 30° and 31°; one would interpolate to arrive at a reasonable estimate. The most common interpolation techniques are finite difference methods; to improve precision, more known values are taken on either side of the desired value, and combined into some sort of equation. Originators and improvers on finite difference methods include Bessel, Gauss, Newton, and Stirling.

The solution of systems of linear equations is also in the province of numerical analysis. The solution methods are handled as special cases of matrix operations such as row-reduction, inversion and eigenvalue determination. Linear equations are important in circuit analysis and in the modelling of economic systems. Gauss and Gauss-Jordan elimination methods are commonly used solution methods, while Gauss-Seidel iteration and Graeffe's root squaring method are applied to difficult problems.

Applications to Computer Science

The solution of many physical problems would not be possible without the application of numerical analysis. Indeed, design of computers probably would be prohibitively expensive without numerical analysis on computers. Investigation of algorithms for calculating common functions is also important in the design of mathematical software.

References

Acton 1970
Fike 1968
Hamming 1962
Hildebrand 1974
Householder 1953

IV.A.10. QUEUEING THEORY

General Definition

Queueing theory is the study of queueing systems--systems where customers arrive (usually at random) and await service, and leave after service (perhaps before, if they become impatient); here, "customers" may be people as well as, for example, objects on a production line, or computer programs awaiting execution. Queueing theory can also be used to model waiting line situations, providing parameters such as average waiting time or queue length.

Historical Perspective

The initial motivation for the development of a theory of queues was the study of telephone traffic congestion. Erlang was the pioneer in the field, and in 1901 he published The Theory of Probabilities and Telephone Conversations. Further applications of theory in telephony were made in the 1920's by Molina, and in the early 1930's by Pollaczek. Applications in other areas, such as management science and operations research, have been published by Champernowne on random walks, Karlin and McGregor on birth-death processes, and Takács on waiting time.

Principles and Theories

The characteristics of a queue are usually represented in a notation due to Kendall. The following five queue features are basic to the Kendall notation:

. Arrival time distribution. Is the arrival pattern probabilistic, deterministic, exponential, or of another type?

. Service time distribution. Does the service time vary predictably, is it constant, or does it vary unpredictably?

. Number of parallel servers. How many channels are available for providing service? For example, how many checkstands in a supermarket are manned?

. Restriction on system capacity. Basically, how many customers the waiting room (queue) can accommodate.

. Queue discipline. Examples include first come, first served, service in random order, and service based on priority. Also included is the propensity of customers for switching queues in a multi-server system, or for leaving the system before service if the waiting time is excessive.

Queueing theory applies notions from probability and statistics to the design of queueing models. Poisson and Markov properties of distributions, as well as ergodic theory, are commonly used in such analyses. Sometimes, the properties of a queue cannot be expressed in analytical mathematical terms; then, simulation techniques are employed in efforts to solve queueing problems.

When queueing theory is applied to multistage processes such as assembly lines, and optimization techniques are employed in efforts to reduce costs or waiting times, the techniques used are referred to as scheduling theory. In scheduling theory, an additional system characteristic of importance is the order in which tasks must be performed.

Applications to Computer Science

Queueing theory has computer applications in the design of system monitors which schedule programs for execution, particularly when systems allow time-sharing or multi-processing operation.

References

Gross 1974
Lee 1966
Morse 1958
Prabhu 1965
Takács 1962

IV.A.11. SCHEDULING THEORY

General Definition

There are two disciplines commonly called scheduling theory. The mathematical field deals with multistage queueing processes; the management field is concerned with project planning.

Historical Perspective

Analytical treatment of scheduling problems began only in the 1950's, though it quickly grew with the development of such techniques as linear and dynamic programming. Changes in management notions about scheduling came in the early 1960's, necessitated by great complexity of projects and large budgets for their accomplishment.

Principles and Theories

Mathematical scheduling theory uses queueing theory to develop models of assembly line types of processes. Usually, the analysis is continued in efforts to optimize a scheduling process with respect to time or cost. The optimization techniques used include dynamic programming and other methods for dealing with sequences of processes. One of the most important considerations deals with the necessity of one procedure following another or the possibility of two procedures being performed simultaneously; for example, it makes no sense to tighten the lug nuts on a car wheel before mounting the tire, but different tires can be mounted simultaneously by different people.

In management, scheduling theory is used to plan and schedule the tasks involved in a project. Again, the possibility of two tasks being accomplished simultaneously or the necessity of one task's completion before starting the next are fundamental considerations. Management scheduling theory has developed a set of tools for allocating resources to tasks according to their importance and in order to complete a project in a specified amount of time. Among these are Program Evaluation Review Techniques (PERT) for time and cost, and the Critical Path Method (CPM). PERT-Time involves estimating three parameters for each task: the shortest possible time for completion, the longest possible,

and the most probable; these estimates are then combined to give an idea of how the project should be managed. PERT-Cost works analagously with respect to cost. CPM finds the longest sequence of tasks which will accomplish the project as a guide to progress which should be expected: if any task along the critical path is delayed, the completion of the entire project is delayed.

Applications to Computer Science

Mathematical scheduling theory is used in multiprogramming executives for computer systems. Management scheduling methods can be used in the design and construction of computer systems.

References

Archibald 1967
Ashour 1972
Cleland 1968
Conway 1967
Iri 1969
Johnson 1967
Le Breton 1961
Wagner 1970

IV.A.12. SYSTEMS THEORY

General Definition

Systems theory is the application of mathematical techniques in the theoretical modelling of physical systems. Techniques used come from many mathematical disciplines, including algebra, geometry, calculus, probability and combinatorics.

Historical Perspective

The need for a general systems theory was expressed by Boulding in 1956. The theory must mediate between highly general and theoretical mathematical notions and the specialized ideas of particular scientific disciplines.

There are several approaches to systems theory. Wymore looks upon systems theory as a generalization of automata theory; analysis and topology can then be used to solve problems, whereas they are rarely employed in automata theory. Mesarovic views systems as relations, or connections, between sets of input and sets of output; he also includes notions about the temporal order of and causal links between events. Hammer, in contrast to the aforementioned authors, has attempted to redefine notions in mathematics to allow their broader application in systems theory; one of his main goals is extend classical concepts to apply meaningfully to discrete as well as to continuous cases.

Principles and Theories

Different authors have different formal definitions of "system," but they usually share two characteristics: a distinction (which is often not completely definitive) is made between a system and its environment, and facts about the system are characterized as relevant or irrelevant to the problem. For example, in analysis of a service station, the station, its employees and the automobiles entering the station comprise the system, the rest of the world is the environment, and the neighborhood in which the station is located is in

some ways part of the system (e.g., when lines of cars awaiting service extend onto the roadway), but also part of the environment. Relevant system parameters include service time and time of arrival of a car, whereas the color of the car is probably unimportant. Systems theory may be used to study a system, or to design one having the desired features, the system being either real (e.g., a service station), or abstract (as in linguistics or algebra).

Fundamental concepts about systems include:

- resolution level. How accurately system parameters can be measured; e.g., mass to within 1 gram, velocity to 1 km/sec, etc.
- activity. The variation of all system parameters in a time interval.
- time-dependence of relations. If a relation among system parameters is satisfied for every possible activity over an entire time interval, it is absolutely time-invariant; if it is satisfied over the entire interval for some activities it is relatively time-invariant; and if satisfied at some places on a time interval it is locally time-invariant.
- behavior. A time-invariant relation which obtains in a system is termed the system behavior, and classed as permanent, relatively permanent, or temporary, depending on whether the relation is absolutely, relatively, or locally time-invariant.

. organization. Factors of a system producing its behavior. Usually, the organization is seen as divided into the system structure, which does not change, and the program, which is variable.

Based on the above and other characteristics of a system, systems theory applies mathematical techniques such as queueing theory or probability and statistics to model the system, allowing easier analysis and facilitating predictions about the effects of changes on the system. Another use of systems theory is in the design of a system structure given knowledge of its desired behavior.

Systems theory is also used in simulation of physical systems. Simulation is basically the use of models to produce quantitative estimates about the parameters of a system. Often, simulations must use simplifications of models, since the complete model is of great computational complexity.

Applications to Computer Science

Systems theory can be used to model computer systems, and thus study the costs of time-sharing or parallel processing. Computers are also necessary in the modelling of systems with a great degree of complexity, such as national economies.

References

Hammer 1969
Klir 1969
Meredith 1973
Mesarovic 1975
Padulo 1974
Wymore 1969

IV.B.1. ARTIFICIAL INTELLIGENCE

General Definition

Research in the field of artificial intelligence (AI) encompasses a broad range of problems, all of which involve the imitation of human thought processes. For example, the design of a machine which can play chess, and which learns and improves its game as it plays more, is a possible research problem in AI.

Historical Perspective

Two important philosophical contributions to AI are by Wiener and Turing. Wiener, in Cybernetics, attempts to show what diverse fields must be united in pursuit of AI devices or theory, and he discusses some of the moral implications of AI as well. Turing, in Computing Machinery and Intelligence, proposed a test to answer the question "Can machines think?"

Most research in AI in the 1950's and early 1960's was concentrated on producing machines or programs which perform some task which humans consider "intellectual" activity. Early examples include Samuel's checkers-playing program, Gelernter's geometry theorem-proving machine, the logic theorist of Newell, Shaw, and Simon, Slagle's symbolic integrator, and numerous chess-playing programs. Machines which attempt to understand natural language, and machines which discern and recognize patterns, are also the results of specific AI research.

Another branch of AI is concerned directly with the simulation of human thought processes. Newell and Simon's general problem solver is designed to simulate human thought, and provide insights into the processes which occur between problem statement and problem solution. Other programs simulate human inductive inferences and social behavior.

Modern AI research includes continuation of much of the above-mentioned work, as well as attempts to translate natural languages by machines. A particularly important application is pattern recognition, which is now used by banks and, to a limited extent, by the Postal Service. Improving pattern recognizers could make the completion of document handling tasks much faster and more reliable.

Principles and Theories

One of the primary efforts in AI is aimed at the development of heuristic methods. Heuristic methods are solution attempts which will not always work but which offer a good chance of arriving at the correct answer; in many cases, they are the only ways to solve a problem. For example, a chess-player cannot look ahead and examine all possible moves; so he uses strategies which usually work or have worked in the past in similar situations. A mathematician, in attempting to prove a theorem, can work backward from the result desired, as well as forward from known expressions. Heuristics are powerful tools in problem solving, but also have application in other types of AI. For example, humans derive much information from speech through heuristics about what they expect to hear. If a computer

could be programmed to use these heuristics, it could listen to and comprehend human speech.

Problems about human learning processes--how many times a stimulus-response pair must be repeated before the stimulus elicits the correct response; the mechanisms of memory and forgetting; the formation of general classes or concepts from specific instances of elements of such classes; how humans decide questions based on limited evidence; and many more--are the subject of another branch of AI; proposed solutions in this area run into the problem of psychological reality--how does one know that a simulator is doing its job using the same processes a human would?

Though no general theory of pattern recognition exists, there is a fairly standard way of studying pattern recognition problems. Problems are segmented into three subproblems:

- sensing. How will the machine sense the pattern, and how will it be represented in the machine?
- feature extraction. What characteristics can be used to best distinguish different input data items? For example, what are the salient features of letters of the alphabet which identify them?
- decision algorithm. What boundaries should be established to assure high probability of correct identification? In other words, how many criteria must a letter satisfy to be identified as a particular letter?

Methods of representing the data items expected are also studied in pattern recognition.

Applications Related to Computer Science

Many of the AI projects mentioned above--chess-players, theorem-provers, and pattern-recognizers--would not be implementable without the processing power of computers. AI also has implications for computer design: if a computer had the memory organization of a human, its access time might be reduced, or it might process in parallel much more readily than do today's systems.

References

Andrews 1972
Atkinson 1965
Banerji 1969
Dyer 1966
Feigenbaum 1963
Findler 1971
IBFI 1963
Kanal 1968
Minsky 1973
Pedelty 1963
Sass 1965
Schank 1973
Tou 1974
Watanabe 1969
Watanabe 1972
Wiener 1948
Wilson 1966

IV.B.2. OPTIMIZATION THEORY

General Definition

Optimization theory involves the application of mathematical techniques to the solution of problems which objective is the maximization

or minimization of a function, sometimes subject to constraints. Classical mathematical optimization theory includes methods from differential calculus, multivariate calculus, and the calculus of variations. Modern developments in optimization theory include linear and non-linear programming, game theory, dynamic programming, and elements of control theory.

Historical Perspective

The development of differential calculus by Newton and Leibniz allowed the solution of problems which had been unsolved before. Using techniques from multivariate calculus, Lagrange developed his multiplier method of finding extrema of functions subject to constraints. Euler was the first to state general rules for solving problems in the calculus of variations, while Lagrange introduced much of the terminology of the field.

The mathematical foundations for linear programming were laid by von Neumann. In 1946, Dantzig published the simplex algorithm for solving linear programming problems, and computer programs implementing it were achieved in the early 1950's. Non-linear programming is not as advanced, but major tools include the Cauchy gradient method and the Kuhn-Tucker optimality criterion.

The foundations of game theory were laid in 1921 by Borel, but the minimax theorem (fundamental to game theory) was not proved until 1927, by von Neumann. Widespread attention to game theory came only in 1944, with the publication of a book about it by von Neumann and Morgenstern.

Dynamic programming is an extension of the calculus of variations. Bellman was the first to use the term, and he proved a fundamental theorem in the field.

Principles and Theories

The use of differential calculus in optimization problems classically involves finding points at which a function achieves an extremum; unfortunately, the simple techniques which indicate an extremum for functions of one variable do not readily extend to functions of two or more variables. The method of Lagrange multipliers is used to solve multivariate optimization problems subject to constraints. The calculus of variations involves the optimization of functionals, where a functional is a rule assigning a numerical value to a function; some of the simplest problems are concerned with finding what arc between two points has optimal properties for some function.

Linear programming is a set of methods for finding extrema of linear functions subject to linear constraints. For example, the mixing of quantities of different substances, subject to minimum amounts of each substance in the mixture, and attempting to minimize the manufacturing cost, is a linear programming problem.

Game theoretic analyses involve classification of a game by such features as the number of players or teams and determinations of utility--e.g., the payoff of the game. (Usually, it is presumed that players have conflicting interests--they work at cross-purposes.) Then, an effort is made to determine an optimal strategy--one which will lead to maximum gains (or minimum losses).

Applications of control theory in optimization are mainly concerned with the use of feedback techniques. An example would be an electronic robot limb which accepts commands from a processor and returns information about its location, orientation, pressure exerted on it, etc., so that the processor can direct its movements for best effect.

Dynamic programming is the mathematical theory of multi-stage decision processes. A sample problem: to find the path between two locations in a city which will take the least time. Bellman's fundamental theorem states that if a problem is decomposed into subproblems, then the optimal solution to the total problem is the union of the optimal solutions to each subproblem.

Applications Related to Computer Science

Many optimization problems would be intractable without the use of computers, since they are quite complex. For example, attempting to solve the dynamic programming problem mentioned above could easily involve choosing among thousands of alternate routes in a moderately large city.

References

Bellman 1963
Beveridge 1970
Gottfried 1973
Mangasarian 1969
von Neumann 1953

IV.B.3. PROBABILITY AND STATISTICS

General Definition

Probability theory predicts the results of performing repeated experiments, based on knowledge of the experimental conditions. Statistical

analysis is used in efforts to discover significant patterns in experimental data, and to estimate the meaningfulness of experimental results. Probability and statistics, then, are mathematical disciplines with applications in the design of experiments and the analysis of experimental results.

Historical Perspectives

Intuitive notions about probability have existed for thousands of years, but the first quantitative work in the field was done by Fermat, Huygens, and Pascal in connection with "games of chance"; later work in this area was accomplished by Bernoulli and DeMoivre. Boltzmann and Gibbs applied probability notions in physics, and Wiener applied it in his formal description of Brownian motion. Kolmogorov is responsible for many modern developments in probability.

Statistical methods are largely based on probability. Some of the most important methods have been developed by Bayes, Neyman, and Pearson; these deal with data grouping and class definition: finding patterns in data. Two important applications of statistical analyses are due to Fisher (theoretical considerations in experimental design) and Wald (quality control).

Principles and Theories

Elementary probability uses combinatorial notions; a sample question might concern the number of possible orderings of a card deck. The probability of an event is simply the ratio of the number of ways that event can occur to the total number of possible events: out of all

possible deals of a card deck in a poker game, how many will yield a royal flush?

Statistics about experimental data are often concerned with finding average values, and with fitting experimental data to equations. There are several types of average--mean, median, and mode; the mean is most commonly used but can be deceptive: a city advertising mean temperature 70° could have six months of 30° weather and six months of 110° weather each year. The median is the number at the middle of an ordered list of the data. The mode is that datum which occurs most often. For the set (2,2,5), the mean is 3, the median 2, and the mode 2. When using averages, it is important to know the data distribution (see below).

Probability and statistics deal with random variables as elements of event spaces. An event space is comprised of points representing events. A fundamental concept is the distribution of points in the event space. The distribution refers to the ways in which data may be aggregated or clustered; distributions are classified according to such features as modality, normality, and skewness. For example, the bell-shaped distribution curve is normal and unimodal; this means that the mean, the median, and the mode coincide. An important concept in random distributions is due to Poisson. In advanced probability, the measure theory of Lebesgue is employed, and such studies are sometimes said to be part of the theory of distributions.

Stochastic processes are sequences of probabalistic events. An example is a card game: each deal of a card is random. Markov chains are a special type of stochastic process. The study of random walks--in which the movement of a particle is independent of its past history--is a particularly rich

field for investigation; many physical processes can be modelled on the atomic level by random walks.

Applications Related to Computer Science

Probability and statistics are relevant in such computer science related fields as queueing theory and finite mathematics. Computers are also important in applications of probability and statistics: large statistical analyses could not be carried out without computers, due to the magnitude of the calculations involved. Simulation also relies heavily on theories from probability and statistics.

References

Feller 1968
Hodges 1964
Karlin 1966
Kemeny 1960
Larson 1974
Pfeiffer 1973

V.A.1. CIRCUIT THEORY

General Definition

Circuit (or network) theory is considered today as a component discipline of systems theory and as a mathematically well-developed electrical engineering field. It covers both the determination of what a given circuit will do (analysis) and the design of circuits with specified characteristics (synthesis).

Historical Perspective

Before the first World War, circuit theory had been deduced from Maxwell's general electromagnetic theory, and was viewed as an independent discipline. Much of the progress in circuit theory in the period immediately after World War I was in connection with network design for long distance telephony. In the latter half of the 1920's and the beginning of the 1930's, a transition from methods of circuit analysis to network synthesis occurred. During the second World War, microwave applications of network theory led to the adoption of techniques from physics in preference over classical methods.

After World War II, circuit theory grew in many directions. The theory of active networks progressed, and a separation between formal realizability theory and topological synthesis developed. One important research area involves design of circuits which perform a task faster or more reliably. In this connection, integrated circuit theory has allowed a shift in emphasis from minimization techniques to methods of circuit construction which allow faster operation.

Principles and Theories

Circuit theory takes as postulates Kirchoff's laws: (1) the voltage difference between two points in a circuit is the same no matter how the circuit is traversed; (2) the current entering a junction equals the current leaving; and Ohm's law: the voltage difference across two points equals the product of the current flow and the resistance between the points. Since ordinary circuit elements like resistors and capacitors are analogous to elements of dynamic systems like dashpots and springs, several techniques were adopted from analytical dynamics; Steinmetz, for example, popularized the complex notation for steady state harmonic behavior. Heaviside was among the first to use the concept of impedance, which refers to the tendency of circuit elements to resist alternating electric current.

The invention of the electric filter by Campbell and Wagner was crucial to telephone technology. Zobel contributed to filter techniques with his m-derivation. Filters act on signals by damping certain frequencies and not damping others. They can make communication lines less noisy.

Cauer and Foster contributed the first papers in the area of network synthesis. Later, Butterworth and Cauer applied approximation techniques in solving amplifier and feedback design problems.

The development of transistors and, later, integrated circuits, allowed circuit designers to concern themselves less with minimization techniques--using as few components as possible to do a job, for reasons of cost and reliability--and more with speed and reliability of circuit elements. This new technology has also enabled the reduction in size of circuit components by a factor of several thousand.

Applications to Computer Science

Circuit theory in computer science is used to design the circuits which make up a processor. Some examples include timing and delay circuits, flip-flops or multi-vibrator circuits, and gating circuits for performance of logical operations.

References

Anderson 1973
Chu 1962
Lepage 1952
Marcus 1973
Mavar 1973
Murdock 1970
Newcomb 1968
Van Valkenburg 1974

V.A.2. CRYOGENIC ENGINEERING

General Definition

Cryogenic engineering is the technology of producing low temperatures and the study of phenomena, such as superconductivity, which occur at low temperatures. In this context, a low temperature is near or below the boiling point of methane, around 112 K.

Historical Perspective

Major early accomplishments in cryogenics include the first liquefaction of oxygen by Cailletet and Pictet (independently), in 1877; Olszewski and Wroblewski, in 1883, developed another oxygen liquefaction method and a device for storing the liquid element. These techniques were used in the determination of the components of air.

Important developments around the turn of the century include the liquefaction of large quantities of air by Linde (1895) and Claude (1902); Dewar's liquefaction of hydrogen (1898), and his development of a vacuum-insulated storage vessel; and the liquefaction of helium in 1908 by Onnes.

With gas liquefaction techniques well developed, work began on extending the range of low temperatures available, exploring applications of cryogenics, and study of phenomena occurring at cryogenic temperatures. A notable magnetic cooling method, proposed in 1926 and implemented in 1933, allowed production of temperatures around .001 K. Goddard experimented with liquid oxygen as a rocket fuel in the 1920's.

Onnes discovered the phenomenon of superconductivity (the disappearance of resistance to electricity in metals at cryogenic temperatures) in 1911. In 1933, Meissner and Ochsenfeld showed that superconductive states also completely exclude magnetic flux. Study of the microscopic characteristics of the superconducting state led to the distinction of Types I and II superconductors; the former are usually pure metals, while the latter are alloys. Bardeen, Cooper, and Schrieffer first gave a complete microscopic explanation of superconductivity, for which they won a Nobel prize.

Principles and Theories

The major area of interest in cryogenic engineering is the development of methods for achieving and maintaining low temperatures. To this end, much work has been done in areas such as heat transfer and insulation;

the former is of interest because large amounts of heat-generating energy must be expended to cool to cryogenic temperatures, and the latter is important for the maintenance of low temperatures once achieved. Almost all methods for gas liquefaction are based on the Joule-Thomson effect of isenthalpic expansion--if an insulated container of gas is allowed to expand, the gas temperature decreases.

In superconductivity, interest has been in finding good superconductors, in the effects of various physical properties of a metal on its superconducting behavior, and in generation of large magnetic fields using superconducting magnets.

The microscopic study of superconductivity involves investigations into transition states between low resistance and zero-resistance states, as well as determination of the structure of a superconducting metal or alloy. One particularly interesting microscopic process is tunnelling, discovered by Giaever in 1960; electrons can cross a gap (tunnel) between superconducting metals or between a normal metal and a superconductor, if a voltage difference exists between them.

Applications to Computer Science

A major application of cryogenic properties to computer science is the Josephson junction, a circuit which is fabricated from superconductors and which operates at extremely high switching speeds and with very little power dissipation, exceeding by far the characteristics of any known transistor.

References

DeGennes 1966
Fishlock 1969
Haselden 1971
Parks 1969
Rose-Innes 1969
Wallace 1968
Williams 1970

V.A.3. DISPLAY SYSTEMS ENGINEERING

General Definition

Display systems engineering is the application of physical science and technology to the design of devices which display data.

Historical Perspective

Devices for data display have existed for centuries; a simple example is a clock. But with the tremendous increase in information, brought about largely by the digital computer, a need has arisen for "better" information display methods--better in the sense of more easily comprehensible or more efficiently presented. A modern trend in computer display systems is toward devices which themselves process data, instead of allowing the computer to do all of the processing.

Principles and Theories

Three elements of display systems are easily distinguished:

. initial transducer. This element receives the information to be displayed.

- intermediate transducer. This element transforms the "raw" data received by the initial transducer into another form.
- depiction mechanism. This element accepts the transformed data from the intermediate transducer and gives human-understandable (e.g., auditory or visual) output.

For example, in a modern watch, vibrations of a quartz crystal are counted and transformed into electrical impulses which excite light-emitting diodes to indicate the time of day or the date.

Initial transducer design depends very much on the application. Common initial transducers include terminal keyboards and card punch machines. Radar antennae, spectrophotometers, and voltmeters are possible initial transducers in different specialized applications.

Intermediate transducers may be digital computers devoted to the task or special purpose devices which incorporate logic circuitry. Many graphic display terminals, for example, have microprocessors which accept information from a computer and from controls on the terminal; the processor then performs necessary transformations to allow viewing of a figure from different perspectives.

The design of depiction mechanisms for optimum comprehension is based on psychological factors, and implementation uses techniques from photometry, colorimetry, optics, and electronics. For example, an engineering

psychologist might be called upon to determine, for a cathode-ray display, the best character size or the best color contrast with the background; it would then be the engineer's task to build such a device, subject to constraints on image quality, possible viewing angle, and so on, as well as to limitations inherent in the materials used for construction.

Applications to Computer Science

Display systems in computer science are the user's way of determining what the computer is doing, and are particularly important in interactive computer use. Specialized display systems are also used in engineering design; some allow "blueprints" to be stored in a computer and modified using cathode-ray devices, without necessitating costly manual redrawing; others use computers to design better computers.

References

Barnhill 1974
Biberman 1971
Biberman 1973
Luxenberg 1968
Sherr 1970
Woodworth 1967

V.A.4 SIGNAL PROCESSING

General Definition

Signal processing is concerned with methods of collecting and analyzing signals. A signal is an electrical quantity which corresponds to physical data; for example, if an anemometer drives a magneto, the power output signal will vary with the speed of revolution of the anemometer, and hence with the windspeed.

Historical Perspective

Analog signal processing dates to the period immediately after the first World War, with research in long distance telephony, particularly in filter design; but such research was considered to be a part of circuit theory. Speculation about the application of digital techniques to filter design can be traced back to World War II, but a formal theory of digital signal processing did not emerge until the middle of the 1960's, when reliable integrated circuits were available. Kaiser showed how to design a digital filter. Cooley and Tukey published a fast method of computing discrete Fourier transforms. Stockham demonstrated that finite impulse response filters could be very efficient.

Principles and Theories

There are five major areas of interest in signal processing:

. data capture and storage. This area is concerned with the mechanisms for collecting data and for translating

it into signals, as well as with devices which store the information.

. pre-processing. Often, data as initially received is not readily analyzable. For example, in the decay of a radioactive sample, the emission of particles is sporadic, but the figure of interest is particles emitted per unit time; so, a pre-processor would store up the number of particles detected and send the information out at regular time intervals. Pre-processing may also involve conversions between analog and digital data forms.

. frequency analysis. This topic is concerned with expressing signals in terms of their amplitude or power, the location of such parameters on a frequency spectrum, and the problem of studying infinite series of random signals in terms of statistics acquired in a finite time interval.

. filtering techniques. Due to non-idealities in signal generators, signals often need to be modified to be processed; a filter accepts discrete inputs and produces output dependent upon these inputs and on desired characteristics of the filter.

. correlation analysis. This area is based heavily on statistics and is used to determine similarity between signals and to separate signals from noise.

Applications to Computer Science

Signal processing is important in the design of reliable computer circuitry, and in construction of transducers which translate data into signals. Computers are now used in the design of signal processors, and thus can aid in the design of new computers.

References

Beauchamp 1973
Gold 1969
Martin 1969
Rabiner 1975
Schwartz 1975
Whalen 1971

V.B.1. COMPUTER STORAGE TECHNOLOGY

General Definitions

Computer storage technology is concerned with the research, development, and production methods for computer memory devices. It uses theory from chemistry and physics, and techniques from engineering, in accomplishing this task.

Historical Perspective

The need for computer storage devices is a concomitant of the development of computers. Slow mechanical memory devices such as those used in mechanical calculators existed before modern computers, but the fast processing capability of modern computers made the development of faster memories important; thus, researchers turned to ferromagnetic and other advanced technologies in search of faster storage devices. Early work in the late 1940's and early 1950's used ferromagnetic devices like cores and drums, as well as semiconductor devices, including the transistor. In the 1960's, integrated circuits further improved computer memory performance. In the 1970's, technologies being explored for use as computer memories include magnetic "bubble" devices, charge-coupled devices, and holography.

Principles and Theories

Main memory in a computer contains programs being executed and their associated data. It is fast to read from and write into, and the data in each memory location can be obtained in the same amount of time (access time). The capacity of main memory is usually small because of its high cost. Main memories are often volatile: interruptions in power cause loss of the

information stored therein. Common main memory implementation methods include integrated circuits and other semiconductor devices; magnetic core memories are also used, and they are not volatile.

Bulk memory devices such as magnetic tapes and disks store programs for future use. They are capable of storing more data than main memory, and they have longer access time. In addition, they are not volatile. A common bulk memory design uses ferromagnetic materials coated onto a substrate such as plastic or aluminum. The ferromagnetic coating material stores binary information by being magnetized in ways which correspond to binary "one" and "zero".

In memory devices using ferromagnetic films, there are two types of accessibility: sequential and random. Random access devices, such as disks and drums, have heads which float on a film of air above the media surface; these heads read or write data and there are mechanisms which control movement of the head to insure that data operations occur at the correct locations. Sequential access devices, such as magnetic tapes, have stationary read and write heads in contact with the moving tape. Mechanisms are needed to prevent tape breakage during the sudden starts and stops made.

Experimental devices using holographic techniques, magnetic "bubbles," charge-coupled devices, and other media, are being explored for possible use in faster or larger capacity[1] memories. Reduced cost is also a goal.

---

[1]Larger capacity can be achieved either by increasing device size or by increasing the packing density--how much data is stored per unit area.

Applications to Computer Science

Memory is a vital part of a computer, without which modern applications would be impossible. It is used to store the large amounts of data used by computer programs, as well as the programs themselves. The speed of memory devices affects the speed with which computations can be performed.

References

Eimbinder 1971
Flores 1973
Hoagland 1973
Hodges 1972
Hodges 1973
Laliotis 1973
Luecke 1973
Marcus 1973
Pear 1967

V.B.2. MICROPROGRAMMING

General Definitions

Microprogramming is a technique for implementing a computer's control function using memory.

Historical Perspective

Wilkes coined the term "microprogramming" in 1951, describing computer instructions which perform many transfers of data in one cycle. Implementation of microprogramming did not become popular, however, until the development of fast reliable integrated circuits in the mid-1960's.

Principles and Theories

In microprogramming, microinstructions reside in control storage, which is always a fast memory and is usually faster than main memory (Here, the speed of a memory refers to the time needed to access parts of it.). This memory may be modifiable only by the computer manufacturer, (e.g., read-only memory - ROM), it may be user modifiable using special techniques (e.g., programmable read-only memory - PROM), or it may be writable by a user in the course of a program, though typically the writing operation is slower than reading.

Microprogramming is implemented using microinstructions. In the design of microinstructions, two major considerations are:

. number of resources controlled by the instruction. A vertical instruction controls single operations - load, store, etc. - on one or more operands; a horizontal instruction controls resources which can operate simultaneously and independently - e.g., main memory, the arithmetic and logic unit, and various registers. A technique known as residual control is a hybrid of vertical and horizontal instruction schema, in which special registers, in conjunction with the instruction, direct computer operation.

. encoding method. Direct encoding means that individual bits control different resources and are combined into instructions; with indirect encoding, the meaning of one field depends on another (control) field, on the value in a register, or on the machine status.

Vertical microinstructions are most often used, being more closely akin to traditional programming methods. Horizontal microinstructions have the potential for being much faster, but are more difficult for the programmer to use.

An important feature of microinstructions is the possibility of overlapped instruction execution, i.e., while one instruction is being executed, the next is fetched for execution. Of course, situations like conditional statements require special handling, since one does not know what the next instruction is until the condition has been evaluated.

Applications to Computer Science

Many modern computers are microprogrammed; in some, a user can design his instruction set to contain only the instructions he needs, allowing faster execution. Microprogramming has been used in graphics systems and for signal processing, as well as in "emulation" of machines by other machines.

References

Agrawala 1974
Falk 1974
Husson 1970
Kampe 1960
Laliotis 1974
Ramamoorthy 1974
Rosin 1969
Wilkes 1953

V.B.3. OPTICAL COMPUTING

General Definition

Optical computing is based on principles from Fourier optics and holography. A wide variety of optical computing applications exists, including image enhancement and pattern recognition.

Historical Perspective

The use of Fourier optics in image manipulation began around the turn of the twentieth century; the first reported experiments of this type were performed by Abbe and Porter. In 1935, Zernike proposed a phase-contrast microscopic technique for observing transparent specimens. In the early 1950's, Maréchal began using spatial-filtering techniques in image enhancement. Stroke and Zech first described the use of "holographic Fourier-transform division image-deblurring convolution" in 1967. The application of optical techniques in radar was also introduced around this time.

Principles and Theories

Fourier analysis is a mathematical tool which is used to determine the frequency spectrum of a function over a surface in terms of integrals of exponential functions. This operation is known as a Fourier transform. Many theorems about the properties of Fourier transforms exist; one, the convolution theorem, states that convolving two functions is equivalent to multiplying their transforms; another, the Fourier integral

theorem, states that applying the transform and then the inverse transform to a function yields the original function.

A lens naturally performs Fourier transforms on images in terms of their spatial frequency. This is the principle underlying holography, wherein monochromatic light (usually laser light) passing through a lens, illuminating an image, produces a hologram in the focal plane of the lens; this hologram can then be reconstructed to give a three-dimensional image. A lens can be used to deblur a photograph: if one regards the picture as one function and the blurring as another, the blurred photograph is a convolution of the two; deconvolving them involves a simple application of the convolution theorem. This same idea can be used to remove grid lines from a picture.

Three important applications of optical computing are in pattern recognition, radar, and image enhancement. Pattern recognition is a particularly fruitful application because of the natural capability of the lens for parallel processing. Information can be stored as its Fourier transform, and patterns can be analysed in terms of their correspondence to this information. The analysis is facilitated by the ability of the lens to search for the same pattern in different areas of a picture (simply shifting the axes of the transform).

Radar antennae located on a moving vehicle send back data which can be processed optically to give an image having high resolution. This technique can be used to map inaccessible terrains on the earth or other plants. Devices incorporating image enhancement can be used to deblur

photographs whose clarity has been compromised by camera movement or other factors. Electron micrographs can also be resolved using these techniques.

Applications to Computer Science

Optical computing is being used in the development of alternative types of input devices, memories, and logic units. For example, a content-addressable memory could be realized using optical computing techniques.

References

Pollock 1963
Rosenfeld 1969
Shulman 1970
Stroke 1972
Tippett 1965

V.B.4. SOFTWARE ENGINEERING

General Definition

Software engineering represents an effort to state general principles to be observed in software design. Hence, a software engineer must be familiar with such computer-related areas as systems architecture, data structuring techniques, programming language characteristics, and the theory of algorithms.

Historical Perspective

Software development has been characterized by several negative features, for example: lack of discipline, in that processes are re-invented for each software project; lack of an adequate specification method for performance requirements, causing extra costs since already written software must be modified or discarded; lack of standard tools for design and verification of software, forcing systems programmers to use unreliable subjective judgements in the evaluation of software performance; and, lack of transferability of software between different computer systems, necessitating extra work on or re-development of a system being transferred from one computer to another.

Software engineering represents an attempt to remedy the problems mentioned above, by characterizing and explicating or developing principles for good software development. Popular interest in software engineering was spurred by NATO-sponsored conferences at the end of the 1960's. A variety of techniques for improving the development of software have been the subject of experiments by computer firms.

Principles and Theories

Software is the set of instructions which direct a computer to perform a specific function. Software is often used to implement the operating system, programming language compilers and interpreters, and utility routines of a general-purpose computer system. Software is also used to implement specialized computer applications in the form of programs.

The practice of software engineering involves five major steps:

. the definition of the purpose of a system -- what it is to do;

. the design of a structure which will fulfill the system's defined purpose;

. the specification of an implementation method, typically a programming language;

. the expression of the structure in the implementation by coding and debugging; and

. the modification of the system to meet goals in performance.

Certain principles guide software engineering, including:

. the notion of modularity: splitting a goal into independent subgoals which when combined will accomplish the intended system function;

. the idea of concealing from a user system features which are unimportant to him; for example designing the software to allocate space to variables, rather than concerning the user with that process;

. the assurance of completeness: that a system does all that it is supposed to do.

The goals of software engineering are system understandability, reliability, efficiency and modifiability. To this end, such practices as structured programming, top-down system design, and program production libraries have been used by various computer firms in large software projects.

Applications to Computer Science

Software engineering is used in the design of computer programs, particularly operating systems. Although such systems will almost always contain errors, the goal of software engineering is the minimization of the importance and occurrence of those errors.

References

Bratman 1975
Dahl 1972
McGowan 1975
Ross 1975
Tou 1970
Weinberg 1971

V.B.5. SYSTEMS ARCHITECTURE

General Definition

Computer systems architecture is the study and design of methods for implementing the logical control function which manages the resources of a computer system.[1]

Historical Perspective

There are basically three types of computer system components: hardware, software, and firmware. It is the job of the systems architect to decide which control functions will be implemented in

---

[1] This definition, proposed to the IEEE Committee on Computer Architecture has been paraphrased from Abrams 1973.

hardware, which in software, and which (if any) in firmware. Important considerations include the speed of operations (in which hardware has an advantage) and the system modifiability (in which software has a great advantage). Hardware includes:

- central processing unit (CPU). The CPU of a computer includes the circuits which comprise the arithmetic and logic unit, the control unit, and the registers used in computations.
- memory. Among the types of memory are "core" or main memory, disk or random-access memory, and tape or sequential access memory. Important considerations in weighing the merits of memory systems are speed of access to desired data, and volatility or permanence of the memory.
- input/output devices. Machines used for communication between the user and the processor; included are card readers and punches, line printers and cathode-ray display tubes.
- communication devices. Machines for controlling the transfer of information from one part of a computer complex to another, including buffer units for moving data between units which operate at different speeds.

Software includes routines and programs stored in the computer which give it instructions for performing tasks. The term software usually includes such routines as programming language compilers and interpreters, utility programs such as editors, linkers, and loaders, and specialized routines for performing mathematical or business-oriented tasks.

Firmware describes control functions which are implemented by micro-programming; that is, using fast memory for the storage and execution

of control tasks.

The design of the executive processor and its associated routines is of fundamental importance in modern computer operations. The executive schedules jobs for execution, secures files against tampering, and keeps account of systems use. Associated routines control input/output operations and interrupt conditions. One of the most important jobs performed by an executive is keeping track of what hardware is operational and what limitations should be imposed on it; for example, if a disk has a blemish which interferes with reading or writing, the executive (or one of its associated routines) must prevent use of that portion of the disk.

Many of the problems faced by a systems architect involve trade-offs; for example, higher security versus speed of operation and memory available for use by the processor; or, cost versus speed of input/output devices. In such cases, the architect must carefully weigh the relative advantages and disadvantages of proposed system designs, in light of the expected application.

Applications to Computer Science

Systems architecture is vital to the design of computer systems if they are to be satisfactory for the application and economically feasible. A particularly important application is executive design to allow multi-processing.

References

Abrams 1973
Amdahl 1964
Beizer 1971
Bell 1971
Donovan 1972
Foster 1970
Gear 1974
Katzan 1973

VI. Bibliography

Abrams 1973 — Abrams, Marshall D. and Stein, Philip G., Computer Hardware and Software: An Interdisciplinary Introduction. Reading, MA: Addison-Wesley, 1973.

Abramson 1963 — Abramson, Norman, Information Theory and Coding. New York: McGraw-Hill, 1963.

Acton 1970 — Acton, Forman S., Numerical Methods That Work. New York: Harper and Row, 1970.

Agrawala 1974 — Agrawala, A. K. and Rauscher, T. G., "Microprogramming: Perspectives and Status," in Institute of Electrical and Electronics Engineers Transactions on Computers (August, 1974), Volume C23, Number 8, pp. 817-837.

Aho 1973 — Aho, Alfred V., Hopcroft, John E., and Ullman, Jeffrey D., The Design and Analysis of Computer Algorithms. Reading, MA: Addison-Wesley, 1973.

Amdahl 1964 — Amdahl, G. M., Blaauw, G. A., and Brooks, F. P., Jr., "Architecture of the IBM System/360," in IBM Journal of Research and Development (April, 1964) Volume 8, Number 2, pp. 87-101.

Anderson 1973 — Anderson, Brian D. O. and Vongpanitlerd, Sumeth, Network Analysis and Synthesis: A Modern Systems Theory Approach. Englewood Cliffs, NJ: Prentice-Hall, 1973.

Andrews 1972 — Andrews, Harry C., Introduction to Mathematical Techniques in Pattern Recognition. New York: Wiley-Interscience, 1972.

Anton 1974 — Anton, Howard and Kolman, Bernard, Applied Finite Mathematics. New York: Academic Press, 1974.

Arbib 1964 — Arbib, Michael A., Brains, Machines and Mathematics. New York: Mcgraw-Hill, 1964.

Archibald 1967 — Archibald, R. D. and Villoria, R. L., Network-Based Management Systems (PERT/CPM). New York: Wiley, 1967.

Ash 1967 — Ash, Robert B., Information Theory. New York: Wiley-Interscience, 1967.

Ashour 1972 — Ashour, Said, Sequencing Theory, New York: Springer-Verlag, 1972.

Atkinson 1965 — Atkinson, Richard C., Bower, Gordon H., and Crothers, Edward J., An Introduction to Mathematical Learning Theory. New York: Wiley, 1965.

Banerji 1969 — Banerji, Ranan B., Theory of Problem Solving: An Approach to Artificial Intelligence. New York: American Elsevier, 1969.

Bar-Hillel 1964 — Bar-Hillel, Yehoshua, Language and Information: Selected Essays on Their Theory and Application. Reading, MA: Addison-Wesley, 1964.

Barlow 1965 — Barlow, Richard E. and Proschan, Frank, Mathematical Theory of Reliability. New York: Wiley, 1965.

Barnhill 1974 — Barnhill, Robert E. and Riesenfeld, Richard F., editors, Computer Aided Geometric Design. New York: Academic Press, 1974.

Bazovsky 1961 — Bazovsky, Igor, Reliability Theory and Practice. Englewood Cliffs, NJ: Prentice-Hall, 1961.

Beauchamp 1973 — Beauchamp, K. G., Signal Processing Using Analog and Digital Techniques. New York: Wiley, 1973.

Beizer 1971 — Beizer, Boris, The Architecture and Engineering of Digital Computer Complexes, Volumes 1 and 2. New York: Plenum Press, 1971.

Bell 1971 — Bell, C. Gordon and Newell, Allen, Computer Structures: Readings and Examples. New York: McGraw-Hill, 1971.

Bellman 1963 — Bellman, Richard, editor, Symposium on Mathematical Optimization Techniques. Berkeley, CA: University of California Press, 1963.

Bellman 1967 — Bellman, Richard E., Linear Equations and Quadratic Criteria, Volume 1 of Introduction to the Mathematical Theory of Control Processes. New York: Academic Press, 1967.

Bellman 1971 — Bellman, Richard E., Nonlinear Processes, Volume 2 of Introduction to the Mathematical Theory of Control Processes. New York: Academic Press, 1971.

Berman 1972 — Berman, Gerald and Fryer, K. D., Introduction to Combinatorics. New York: Academic Press, 1972.

Beveridge 1970 — Beveridge, Gordon S. G. and Schechter, Robert S., Optimization: Theory and Practice. New York: McGraw-Hill, 1970.

Biberman 1971 — Biberman, Lucien M. and Nudelman, Sol, editors, Photo Electronic Imaging Devices, Volumes 1 and 2. New York: Plenum Press, 1971.

Biberman 1973 — Biberman, Lucien M., Perception of Displayed Information. New York: Plenum Press, 1973.

Bobrow 1974 — Bobrow, Leonard S. and Arbib, Michael A., Discrete Mathematics: Applied Algebra for Computer and Information Science. Philadelphia, PA: W. B. Saunders, 1974.

Book 1973 — Book, Ronald V., "Topics in Formal Language Theory," in Aho, Alfred V., editor, Currents in the Theory of Computing. Englewood Cliffs, NJ: Prentice-Hall, 1973.

Boole 1847 — Boole, George, An Investigation into the Laws of Thought, On Which Are Founded the Mathematical Theories of Logic and Probabilities. New York: Dover, "First American Printing."

Borevich 1966 — Borevich, Z. I. and Shofarevich, I. R., Number Theory. New York: Academic Press, 1966.

Bratman 1975 — Bratman, Harvey and Court, Terry, "The Software Factory," in Computer (May, 1975), Volume 8, Number 5, pp. 28-37.

Brillouin 1962 — Brillouin, Leon, Science and Information Theory. New York: Academic Press, 1962.

Chomsky 1957 — Chomsky, Noam, Syntactic Structures. The Hague: Mouton, 1957.

Chomsky 1965 — Chomsky, Noam, Aspects of the Theory of Syntax. Cambridge, MA: Massachusetts Institute of Technology Press, 1965.

Chorafas 1966 — Chorafas, Dimitris A., Control Systems Functions and Programming Approaches. New York: Academic Press, 1966.

Chu 1962 — Chu, Yaohan, Digital Computer Design Fundamentals. New York: McGraw-Hill, 1962.

Cleland 1968 — Cleland, David I. and King, William R., Systems Analysis and Project Management. New York: McGraw-Hill, 1968.

Conway 1967 — Conway, Richard W., Maxwell, William L., and Miller, Louis W., Theory of Scheduling. Reading, MA: Addison-Wesley, 1967.

Dahl 1972 — Dahl, Ole-Johan, Dijkstra, E. W., and Hoare, C. A. R., Structured Programming. New York: Academic Press, 1972.

Davis 1958 — Davis, Martin, Computability and Unsolvability. New York: McGraw-Hill, 1958.

De Gennes 1966 — De Gennes, Pierre G., Superconductivity of Metals and Alloys. New York: W. A. Benjamin, 1966.

Donovan 1972 — Donovan, John J., Systems Programming. New York: McGraw-Hill, 1972.

Dyer 1966 — Dyer, Ralph, et alii, Optical Scanning for the Business Man. New York: Hobbs, Dorman and Company, 1966.

Eilenberg 1974 — Eilenberg, Samuel, Automata, Languages, and Machines. New York: Academic Press, 1974.

Eimbinder 1971 — Eimbinder, Jerry, editor, Semiconductor Memories. New York: Wiley-Interscience, 1971.

Falk 1974 — Falk, Howard, "Microcomputer Software Makes Its Debut," in Institute of Electrical and Electronics Engineers Spectrum (October, 1974), Volume 11, Number 10, pp. 78-84.

Feigenbaum 1963 — Feigenbaum, Edward A. and Feldman, Julian, editors, Computers and Thought. New York: McGraw-Hill, 1963.

Feinstein 1958 — Feinstein, Amiel, Foundations of Information Theory. New York: McGraw-Hill, 1958.

Feller 1968 — Feller, William, An Introduction to Probability Theory and Its Applications, Volumes 1 and 2. New York: Wiley, 1968.

Fike 1968 — Fike, C. T., Computer Evaluation of Mathematical Functions. Englewood Cliffs, NJ: Prentice-Hall, 1968.

Findler 1971 — Findler, N. V. and Meltzer, Bernard, Artificial Intelligence and Heuristic Programming. Edinburgh: Edinburgh University Press, 1971.

Fishlock 1969 — Fishlock, David, editor, A Guide to Superconductivity. New York: American Elsevier, 1969.

Flegg 1964 — Flegg, H. Graham, Boolean Algebra and Its Applications. New York: Wiley, 1964.

Flores 1973 — Flores, Ivan, Peripheral Devices. Englewood Cliffs, NJ: Prentice-Hall, 1973.

Foster 1970 — Foster, Caxton C., Computer Architecture. New York: Van Nostrand Reinhold, 1970.

Fuchs 1971 — Fuchs, Walter R., Cybernetics for the Modern Mind. New York: Macmillan, 1971.

Gauss 1966 — Gauss, Karl Friedrich, Disquisitiones Arithmeticae. New Haven, CN: Yale University Press, 1966.

Gear 1974 — Gear, C. William, Computer Organization and Programming. New York: McGraw-Hill, 1974.

Gold 1969 — Gold, Bernard and Rader, C. W., Digital Processing of Signals. New York: McGraw-Hill, 1969.

Gottfried 1973 — Gottfried, Byron S. and Weisman, Joel, Introduction to Optimization Theory. Englewood Cliffs, NJ: Prentice-Hall, 1973.

Gross 1974 — Gross, Donald and Harris, Carl M., Fundamentals of Queueing Theory. New York: Wiley, 1974.

Gumowski 1968 — Gumowski, Igor and Mira, C., Optimization in Control Theory and Practice. London: Cambridge University Press, 1968.

Hammar 1968 — Hammar, Peter L. and Rudeanu, Sergu, Boolean Methods in Operations Research and Related Areas. New York: Springer-Verlag, 1968.

Hammar 1969 — Hammer, Preston C., editor, Advances in Mathematical Systems Theory. University Park, PA: Pennsylvania State University Press, 1969.

Hamming 1962 — Hamming, Richard W., Numerical Methods for Scientists and Engineers. New York: McGraw-Hill, 1962.

Haselden 1971 — Haselden, G. G., editor, Cryogenic Fundamentals. New York: Academic Press, 1971.

Hildebrand 1974 — Hildebrand, Francis B., Introduction to Numerical Analysis. New York: McGraw Hill, 1974.

Hoagland 1973 — Hoagland, Albert S., "Mass Storage: Past, Present and Future," in Computer (September, 1973), Volume 6, Number 9, pp. 28-33.

Hodges 1964 — Hodges, J. L., Jr. and Lehmann, E. L., Basic Concepts of Probability and Statistics. San Francisco, CA: Holden-Day, 1964.

Hodges 1972 — Hodges, David A., editor, Semiconductor Memories. New York: Institute of Electrical and Electronics Engineers Press, 1972.

Hodges 1973 — Hodges, David A., "Alternative Component Technologies for Advanced Memory Systems," in Computer (September, 1973), Volume 6, Number 9, pp. 34-37.

Hohn 1966 — Hohn, Franz E., Applied Boolean Algebra. New York: Macmillan, 1966.

Hopcroft 1969 — Hopcroft, John E. and Ullman, Jeffrey D., Formal Languages and Their Relation to Automata. Reading, MA: Addison-Wesley, 1969.

Householder 1953 — Householder, Alston S., Principles of Numerical Analysis. New York: McGraw-Hill, 1953.

Husson 1970 — Husson, S. S., Microprogramming Principles and Practices. Englewood Cliffs, NJ: Prentice-Hall, 1970.

IBFI 1963 — International Business Forms Industry, Optical Character Recognition and the Years Ahead. Elmhurst, IL: Business Press, 1963.

Iri 1969 — Iri, Masao, Network Flow, Transportation and Scheduling: Theory and Algorithms. New York: Academic Press, 1969.

Johnson 1967 — Johnson, Richard A., Kast, Fremont E., and Rosenzweig, James E., The Theory and Management of Systems. New York: McGraw-Hill, 1967.

Kampe 1960 — Kampe, Thomas W., "The Design of a General-Purpose Microprogram Computer with Elementary Structure," in Institute of Radio Engineers Transactions (June, 1960), Volume 2, Number EC-9, pp. 208-213.

Kanal 1968 — Kanal, Laveen N., editor, Pattern Recognition. Washington, DC: Thompson Book, 1968.

Karlin 1966 — Karlin, Samuel, A First Course in Stochastic Processes. New York: Academic Press, 1966.

Katzan 1973 — Katzan, Harry, Jr., Operating Systems: A Pragmatic Approach. New York: Van Nostrand Reinhold, 1973.

Kemeny 1960 — Kemeny, John G. and Snell, J. Laurie, Finite Markov Chains. Princeton, NJ: Van Nostrand, 1960.

Kemeny 1966 — Kemeny, John G., Snell, J. Laurie, and Thompson, Gerald L., Introduction to Finite Mathematics. Englewood Cliffs, NJ: Prentice-Hall, 1966.

Kirk 1970 — Kirk, Donald E., Optimal Control Theory: An Introduction. Englewood Cliffs, NJ: Prentice-Hall, 1970.

Klír 1966 — Klír, Jiří and Seidl, Lev K., Synthesis of Switching Circuits. London: Icliffe Books, 1966.

Klir 1969 — Klir, George J., An Approach to General Systems Theory. New York: Van Nostrand Reinhold, 1969.

Knuth 1968 — Knuth, Donald E., Fundamental Algorithms, Volume 1 of The Art of Computer Programming. Reading, MA: Addison-Wesley, 1968.

Knuth 1969 — Knuth, Donald E., Seminumerical Algorithms, Volume 2 of The Art of Computer Programming. Reading, MA: Addison-Wesley, 1969.

Knuth 1973 — Knuth, Donald E., Sorting and Searching, Volume 3 of The Art of Computer Programming. Reading, MA: Addison-Wesley, 1973.

Korfhage 1974 — Korfhage, Robert R., Discrete Computational Structures. New York: Academic Press, 1974.

Kostopoulos 1975 — Kostopoulos, George K., Digital Engineering. New York: Wiley-Interscience, 1975.

Kurki-Suonio 1971 Kurki-Suonio, Reino, A Programmer's Introduction to Computability and Formal Languages. Princeton, NJ: Auerbach, 1971.

Laliotis 1973 Laliotis, Theodore A., "Main Memory Technology," in Computer (September, 1973), Volume 6, Number 9, pp. 20-27.

Laliotis 1974 Laliotis, Theodore A., "Microprocessors Present and Future," in Computer (July, 1974), Volume 7, Number 7, pp. 20-24.

Larson 1974 Larson, Harold J., Introduction to Probability Theory and Statistical Inference. New York: Wiley, 1974.

LeBreton 1971 LeBreton, Preston P. and Henning, Dale, A., Planning Theory. Englewood Cliffs, NJ: Prentice-Hall, 1961.

Lee 1966 Lee, Alec. M., Applied Queueing Theory. London: Macmillan, 1966.

Lee 1972 Lee, John A. N., Computer Semantics: Studies of Algorithms, Processors and Languages. New York: Van Nostrand Reinhold, 1972.

Lefschetz 1965 Lefschetz, Solomon, Stability of Nonlinear Control Systems. New York: Academic Press, 1965.

Lepage 1952 Lepage, Wilbur R. and Seely, Samuel, General Network Analysis. New York: McGraw-Hill, 1952.

Lloyd 1962 Lloyd, David K. and Lipow, Myron, Reliability: Management, Methods, and Mathematics. Englewood Cliffs, NJ: Prentice-Hall, 1962.

Luecke 1973 Luecke, Gerald, Mizer, Jack P., and Carr, William N., Semiconductor Memory Design and Application. New York: McGraw-Hill, 1973.

Luxenberg 1968 Luxenberg, H. R. and Kuehn, Rudolph L., editors, Display Systems Engineering. New York: McGraw-Hill, 1968.

McGowan 1975 McGowan, Clement L. and Kelly, John R., Top-Down Structured Programming Techniques. New York: Petrocelli/Charter, 1975.

Mangasarian 1969 — Mangasarian, Olvi L., Nonlinear Programming. New York: McGraw-Hill, 1969.

Marcus 1967 — Marcus, Solomon, Algebraic Linguistics: Analytical Models. New York: Academic Press, 1967.

Marcus 1973 — Marcus, Abraham and Lenk, John D., Computers for Technicians. Englewood Cliffs, NJ: Prentice-Hall, 1973.

Martin 1969 — Martin, James Thomas, Telecommunications and the Computer. Englewood Cliffs, NJ: Prentice-Hall, 1969.

Mavar 1973 — Mavar, John, M.O.S.T. Integrated Circuit Engineering. Stevenage, England: Peter Peregrinus, 1973.

Meredith 1973 — Meredith, Dale D., Wong, Kam W., Woodhead, Ronald W., and Wortman, Robert H., Design and Planning of Engineering Systems. Englewood Cliffs, NJ: Prentice-Hall, 1973.

Mesarovic 1975 — Mesarovic, Mihajlo D. and Takahara, Yasuhiko, General Systems Theory: Mathematical Foundations. New York: Academic Press, 1975.

Miller 1972 — Miller, Raymond E. and Thatcher, James W., editors, Complexity of Computer Computations. New York: Plenum Press, 1972.

Minsky 1967 — Minsky, Marvin L., Computation: Finite and Infinite Machines. Englewood Cliffs, NJ: Prentice-Hall, 1967.

Minsky 1973 — Minsky, Marvin and Papert, Seymour, Artificial Intelligence: Condon Lectures. Eugene, OR: Oregon State System of Higher Education, 1973.

Morse 1958 — Morse, Philip M., Queues, Inventories and Maintenance. New York: Wiley, 1958.

Murdock 1970 — Murdock, J. B., Network Theory. New York: McGraw-Hill, 1970.

NYU 1970 — New York University Linguistic String Program, An Application of Syntactic Analysis to Information Retrieval. String Program Report Number 6, 1970.

Newcomb 1968 — Newcomb, Robert W., Active Integrated Circuit Synthesis. Englewood Cliffs, NJ: Prentice-Hall, 1968.

Nivat 1973 — Nivat, M., editor, Automata, Languages and Programming. New York: American Elsevier, 1973.

Ollongren 1974 — Ollongren, Alexander, Definition of Programming Languages by Interpreting Automata. New York: Academic Press, 1968.

Padulo 1974 — Padulo, Louis and Arbib, Michael A., System Theory: A Unified State-Space Approach to Continuous and Discrete Systems. Philadelphia, PA: W. B. Saunders, 1974.

Parks 1969 — Parks, R. D., editor, Superconductivity, Volumes 1 and 2. New York: Marcel Dekker, 1969.

Patton 1974 — Patton, P. C., "Data Organization and Access Methods," in Tou, Julius T., editor, Advances in Information Systems Science, Volume 5. New York: Plenum Press, 1974, pp. 1-95.

Pear 1967 — Pear, Charles B., Jr., editor, Magnetic Recording in Science and Industry. New York: Reinhold Publishing, 1967.

Pedelty 1963 — Pedelty, Michael J., An Approach to Machine Intelligence. Washington, DC: Spartan Books, 1963.

Peschon 1965 — Peschon, John, editor, Disciplines and Techniques of Systems Control. New York: Blaisdell, 1965.

Pfeiffer 1973 — Pfeiffer, Paul E. and Schum, David A., Introduction to Applied Probability. New York: Academic Press, 1973.

Pollock 1963 — Pollock, Donald K., Koester, Charles J., and Tippett, James T., Optical Processing of Information, Baltimore, MD: Spartan Books, 1963.

Prabhu 1965 — Prabhu, Narahari, Queues and Inventories: A Study of Their Basic Stochastic Processes. New York: Wiley, 1965.

Rabiner 1975 — Rabiner, Lawrence R. and Gold, Bernard, Theory and Application of Digital Signal Processing. Englewood Cliffs, NJ: Prentice-Hall, 1975.

Ramamoorthy 1974 — Ramamoorthy, Charles V., "A Survey of the Status of Microprogramming," in Tou, Julius T., editor, Advances in Information Systems Science, Volume 5. New York: Plenum Press, 1974, pp. 193-259.

Rose-Innes 1969 — Rose-Innes, Alistair C. and Rhoderick, E. H., Introduction to Superconductivity. Oxford: Pergamon Press, 1969.

Rosenfeld 1969 — Rosenfeld, Azriel, Picture Processing by Computer. New York: Academic Press, 1969.

Rosin 1969 — Rosin, Robert F., "Contemporary Concepts of Microprogramming and Simulation," in Computing Surveys (December, 1969), Volume 1, Number 4, pp. 197-212.

Ross 1975 — Ross, Douglas T., Goodenough, John B., and Irvine, C. A., "Software Engineering: Process, Principles, and Goals," in Computer (May, 1975), Volume 8, Number 5, pp. 17-27.

Salomaa 1973 — Solomaa, Arto, Formal Languages. New York: Academic Press, 1973.

Sass 1965 — Sass, Margo A. and Wilkinson, William D., Computer Augmentation of Human Reasoning. Washington, DC: Spartan Books, 1965.

Schank 1973 — Schank, Roger C. and Colby, Kenneth Mark, editors, Computer Models of Thought and Language. San Francisco, CA: W. H. Freeman, 1973.

Schröder 1966 — Schröder, Ernst, Vorlesungen über die Algebra der Logik. Bronx, NY: Chelsea Publishing, 1966.

Schwartz 1975 — Schwartz, Mischa and Shaw, Leonard, Signal Processing: Discrete Spectral Analysis, Detection, and Estimation. New York: McGraw-Hill, 1975.

Shannon 1956 — Shannon, C. E. and McCarthy, J., editors, Automata Studies. Princeton, NJ: Princeton University Press, 1956.

Sherr 1970 — Sherr, Solomon, Fundamentals of Display Systems Design. New York: Wiley-Interscience, 1970.

Shinners 1972 — Shinners, Stanley M., Modern Control System Theory and Application. Reading, MA: Addison-Wesley, 1972.

Shulman 1970 — Shulman, Arnold Roy, Optical Data Processing. New York: Wiley, 1970.

Slepian 1973 — Slepian, David, editor, Key Papers in the Development of Information Theory. New York: Institute of Electrical and Electronics Engineers Press, 1973.

Stroke 1972 — Stroke, George W., "Optical Computing," in Institute of Electrical and Electronics Engineers Spectrum (December, 1972), Volume 9, Number 12, pp. 24-41.

Szabó 1967 — Szabó, Nicholas S. and Tanaka, Richard I., Residue Arithmetic and Its Applications to Computer Technology. New York: McGraw-Hill, 1967.

Takács 1962 — Takács, Lajos, Introduction to the Theory of Queues. New York: Oxford University Press, 1962.

Tippett 1965 — Tippett, James T., Berkowitz, David A., Clapp, Lewis C., Koester, Charles J., and Vanderburgh, Alexander, Jr., editors, Optical and Electro-Optical Information Processing. Cambridge, MA: Massachusetts Institute of Technology Press, 1965.

Tou 1970 — Tou, Julius T., editor, Software Engineering, Volumes 1 and 2. New York: Academic Press, 1970.

Tou 1974 — Tou, Julius T. and Gonzalez, Rafael C., Pattern Recognition Principles. Reading, MA: Addison-Wesley, 1974.

Traub 1973 — Traub, J. F., Jr., editor, Complexity of Sequential and Parallel Numerical Algorithms. New York: Academic Press, 1973.

van der Waerden 1970 — van der Waerden, B. L., Algebra. New York: Frederick Ungar Publishing, 1970.

Van Valkenburg 1974 — Van Valkenburg, M. E., editor, Circuit Theory: Foundations and Classical Contributions. Stroudsberg, PA: Dowden, Hutchinson and Ross, 1974.

von Neumann 1953 — von Neumann, John and Morgenstern, Oskar, Theory of Games and Economic Behavior. Princeton, NJ: Princeton University Press, 1953.

von Neumann 1966 — von Neumann, John, Theory of Self-Reporducing Automata, edited and completed by Arthur W. Burks. Urbana, IL: University of Illinois Press, 1966.

Wagner 1970 — Wagner, Harvey M., Principles of Management Science with Applications to Executive Decisions. Englewood Cliffs, NJ: Prentice-Hall, 1970.

Wallace 1969 — Wallace, P. R., editor, Superconductivity, Volumes 1 and 2. New York: Gordon and Breach, 1969.

Watanabe 1969 — Watanabe, Satosi, editor, Methodologies of Pattern Recognition. New York: Academic Press, 1969.

Watanabe 1972 — Watanabe, Satosi, editor, Frontiers of Pattern Recognition. New York: Academic Press, 1972.

Weil 1973 — Weil, André, Basic Number Theory. New York: Springer-Verlag, 1973.

Weinberg 1971 — Weinberg, Gerald M., The Psychology of Computer Programming. New York: Von Nostrand Reinhold, 1971.

Whalen 1971 — Whalen, Anthony D., Detection of Signals in Noise. New York: Academic Press, 1971.

Widnall 1968 — Widnall, William S., Applications of Control Theory to Computer Controller Design. Cambridge, MA: Massachusetts Institute of Technology Press, 1968.

Wiener 1948 — Wiener, Norbert, Cybernetics, or Control and Communication in the Animal and the Machine. New York: Wiley, 1948.

Wilkes 1953 — Wilkes, M. V. and Stringer, J. B., "Microprogramming and the Design of the Control Circuits in an Electronic Digital Computer," in Proceedings of the Cambridge Philosophical Society (April, 1953), Volume 69, Part 2, pp. 230-238.

Willems 1970 — Willems, Jacques Leopold, Stability Theory of Dynamical Systems. New York: Wiley-Interscience, 1970.

Williams 1970 — Williams, J. E. C., Superconductivity and Its Applications. London: Pion, 1970.

Wilson 1966 — Wilson, Robert A., Optical Page Reading Devices. New York: Reinhold Publishing, 1966.

Woodworth 1967 — Woodworth, Forrest, Graphical Simulation. Scranton, PA: International Textbook, 1967.

Wymore 1967 — Wymore, A. Wayne, A Mathematical Theory of Systems Engineering - the Elements. New York: Wiley, 1967.

NBS-114A (REV. 7-73)

U.S. DEPT. OF COMM. BIBLIOGRAPHIC DATA SHEET	1. PUBLICATION OR REPORT NO. NBSIR 75- 780	2. Gov't Accession No.	3. Recipient's Accession No.

4. TITLE AND SUBTITLE

MATHEMATICS AND ENGINEERING IN COMPUTER SCIENCE

5. Publication Date: August 1975

6. Performing Organization Code

7. AUTHOR(S)

Christopher J. Van Wyk

8. Performing Organ. Report No.

9. PERFORMING ORGANIZATION NAME AND ADDRESS

NATIONAL BUREAU OF STANDARDS
DEPARTMENT OF COMMERCE
WASHINGTON, D.C. 20234

10. Project/Task/Work Unit No. 6000910

11. Contract/Grant No.

12. Sponsoring Organization Name and Complete Address *(Street, City, State, ZIP)*

Same as item 9

13. Type of Report & Period Covered: Final

14. Sponsoring Agency Code

15. SUPPLEMENTARY NOTES

16. ABSTRACT *(A 200-word or less factual summary of most significant information. If document includes a significant bibliography or literature survey, mention it here.)*

This document presents short descriptions of fifteen mathematical fields and nine engineering fields which are important to computer science today. The descriptions have five sections: general definition, historical perspective, principles and theories, applications to computer science, and references. A list of thirty-five other topics, not separately treated but subsumed under one or more of the twenty-four major headings, is included. A bibliography lists classic and recent works in the areas covered.

17. KEY WORDS *(six to twelve entries; alphabetical order; capitalize only the first letter of the first key word unless a proper name; separated by semicolons)* Computer science; engineering; mathematics

18. AVAILABILITY

☐ Unlimited

☒ For Official Distribution. Do Not Release to NTIS

☐ Order From Sup. of Doc., U.S. Government Printing Office Washington, D.C. 20402, SD Cat. No. C13

☐ Order From National Technical Information Service (NTIS) Springfield, Virginia 22151

19. SECURITY CLASS (THIS REPORT): UNCLASSIFIED

21. NO. OF PAGES

20. SECURITY CLASS (THIS PAGE): UNCLASSIFIED

22. Price

USCOMM-DC 29042-P74


AMERICAN REVOLUTION BICENTENNIAL
1776-1976