report of the

Information Technology Workshop

Robert F. Cotellessa, Chairman

October 1, 1983

Acknowledgments

There are many individuals who contributed to the success of this Workshop and to this report. Special thanks are due to Dr. Lewis G. Mayfield of the National Science Foundation, Dr. Michael Lesk of Bell Laboratories, and Dr. Steven C. Sylvester and Mrs. Margaret Berkery of Stevens Institute of Technology. Dr. Lesk contributed the computer-assisted composition of this report.

REPORT
OF THE
INFORMATION TECHNOLOGY WORKSHOP

Robert F. Cotellessa, Chairman

Under the sponsorship of
Directorate for Engineering
National Science Foundation (NSF)

In collaboration with
Department of Defense

October 1, 1983

INFORMATION TECHNOLOGY WORKSHOP

EXECUTIVE SUMMARY

The purpose of the Workshop, in bringing together research leaders in the information industry, was to meet an important specific recognized need for identifying appropriate areas of research, developing a sense of direction for this research, and developing research agenda for the next 5-10 years in the fields of computer science, computer engineering, and information science.

The Workshop addressed six topical areas, summarized below, in as much depth as possible. Some important topics were not covered and the participants recommended that a subsequent workshop be organized to deal with them.

There were expressed several threads of concern by all of the Workshop groups, as follows:

1 A severe shortage of research manpower exists particularly in universities, in the fields embraced by the Workshop. Significant improvement is not expected during the next several years and the lack of a "critical mass" in some specialties portends undesirably slow research progress. A major government-industry-university effort is needed to deal with this problem.

2 There is still a shortage of science and engineering to serve as a base for applied research and development.

3 There is an increasing need for being able to customize rapidly a computer system with novel architectures for specific areas of research. This need is driven by applications involving words, as well as numbers. The counterpoint to this need is an increased ability to make efficacious use of complex computer systems that can now be constructed.

4 There is a paucity of large-scale computer systems in universities. The lack of access to such systems by university researchers is hindering research progress, both experimentally and theoretically, in all the topical areas which this Workshop addressed

5 The level of research funding is far too low to support adequately the capabilities that exist and the maintenance of a national leadership position.

Research subjects of great importance and promise for achievement in each of the six topical areas addressed are:

1 *Artificial Intelligence*: The recognition of the intent of speakers, the relationships between sentences in continuous discourse, the nonlinguistic aspects of the situation of utterance, and models of the beliefs and goals of the communication agents; interactively and automatically analyzing an existing knowledge base, isolating errors, inference methods, machine learning, machine self-control of problem-solving, the representation of knowledge, and user interfaces.

2 *Distributed Computing and Networking*: Reliability, privacy and security, distribution and sharing, accessing resources and services, distributed databases, network research, design tools and methodology, and theoretical foundations.

3 *Highly Parallel Computing*: Parallel software and the parallel formulation of significant problems, memory and input/output systems for parallel architectures, and the simulation or emulation evaluation of detailed models of proposed architectures.

4 *Information Science*: Simulation of complex categorization algorithms of a highly parallel nature, determination of the critical elements of biological categorization algorithms to guide machine experiments, the role of databases of text and images in adaptive information processing strategies, and the treatment of databases possessing uncertainties and omissions.

5 *Research in Support of Some Major Use Areas*: Language and process development in office automation;

improved methods for manipulating huge numbers of images, alternatives to digital image representation, the attainment of higher processing speeds; software productivity; and large-scale testbeds for computer-aided design and manufacturing.

6 *Robotics*: Languages for specifying robot tasks, the planning of actions and motions, the creation of geometric models of complex objects and environments, universality of robot mechanisms, easily used robot programming systems, better sensors (including multi-image vision systems), locomotion, and computerized replanning techniques.

CONTENTS

INFORMATION TECHNOLOGY WORKSHOP
INTRODUCTION

The frenetic activity world-wide in the field of information technology, in which the United States has occupied a position of leadership, has expanded the horizons of those working in the field and of those who apply, or seek to apply the fruits of a cascade of new achievements. The U.S. position of leadership in information technology is being challenged as never before. The central question that capsules the purpose of the Workshop is:

What areas of research are most important to sustain the vitality of this crucial sphere of national life -- with consequences for the economy and national security?

Since theoretical and experimental research has been the foundation for leadership in this field, the National Science Foundation deemed it desirable for a workshop to be held to meet an important specific recognized need in planning for future research. Thus, the Information Technology Workshop evolved and was sponsored by the National Science Foundation in collaboration with the Department of Defense.

The Workshop brought together invited leaders of the information industry under the sponsorship of the National Science Foundation in collaboration with the Department of Defense. In addressing the expressed need, the Workshop was charged with identifying appropriate areas of research, developing a sense of direction for this research, and developing research agenda for the next 5-10 years in the fields of computer science, computer engineering, and information science. Participants were expected to be mindful of efforts being made in other countries, particularly the Fifth Generation Computer project in Japan. The effort, reported herein, addressed the following six topical areas in as much depth as possible:

- Artificial Intelligence
- Distributed Computing and Networking
- Highly Parallel Computing
- Information Science
- Research in Support of Some Major Use Areas
- Robotics.

Working papers were prepared and distributed in advance of the Workshop. Each of the six groups was given the task of preparing a report that represented a consensus of the participants and was asked to include consideration of the following matters:

An assessment of the state-of-the-art;

An enumeration of problem areas (with respect to current capability and in the realm of theoretical development);

An enumeration of opportunities estimated to be fruitful;

A recommended plan for the next 5-10 years, as well as some more futuristic ideas;

An assessment of the magnitude of human and physical resources that will be required; and

Recommendations concerning the needs for and the mechanisms for university-industry-government efforts.

The Workshop opened with a Keynote Paper delivered by Dr. Jacob T. Schwartz entitled, "Research in Computer Science: Influences, Accomplishments, Goals," followed by a detailed presentation by Dr. Kent K. Curtis on the question, "Computer Manpower - Is There a Crisis?"

The decision as to topics to be addressed, the format of the Workshop, the location and timing of the Workshop, and the development of an invitation list of participants were made by a Steering Committee whose efforts were key to any success achieved.

Areas that were considered for inclusion in the Workshop are:

Machine Intelligence Software Productivity Optical Storage Programming Languages Pattern Recognition and Image Processing Theory.

These topical areas were eliminated, not because of their lack of importance, but to enable the Workshop to perform its task in greater depth. Another Workshop to address these, and possibly other areas, was viewed as desirable.

INFORMATION TECHNOLOGY WORKSHOP
SUMMARY OF FINDINGS AND RECOMMENDATIONS

The six topical areas addressed during this Workshop do not span all those of concern to the information industry. Indeed, the participants expressed the need for a future Workshop be held to deal with several other topics in depth. Research thrusts in the six areas deemed to be of important concern and advantage to pursue during the next 5-10 years were developed and are summarized below, together with statements of need to facilitate this research.

In this undertaking, the maintenance of a leadership position for the U.S. in information technology was a major stimulus; particular cognizance was taken of the Japanese Fifth Generation Computer program. It was generally recognized that a very significant collaborative effort needs to be mounted among government, industry, and universities to achieve this objective.

The findings and recommendations of the six Workshop groups are set forth below.

I. *Artificial Intelligence*

Emphasis was placed on: (1) natural language access to databases and providing output in a natural language; and, (2) expert systems - high-performance programs in complex domains. Successful research will result in such machine characteristics as common sense, the ability to learn from experience, the ability to accept, generate, and act appropriately on natural language input, and perception and general situation assessment. Artificial intelligence (AI) research is central to the Japanese Fifth Generation Computer effort.

In the natural language area, the following recommendations are made:

1 A broad range of basic research support is essential; there is still a shortage of science on which to base natural language system engineering.

2 Novel supercomputer architectures that take AI needs into account are needed; current supercomputers are number crunchers that are of little use or interest to AI.

3 The most active current areas of research, and the most promising for new breakthroughs, are the recognition of the intent of speakers and the relationships between sentences in continuous discourse, taking into account the structure of the preceding discourse, the non-linguistic aspects of the situation of utterance, and models of the beliefs and goals of the communication agents.

In the expert systems area, the following recommendations are made:

1 Near-term research problems on knowledge acquisition and learning include:

a interactively and automatically analyzing and validating an existing knowledge base;

b isolating errors.

Longer-term research includes:

a inferring new inference rules from problem solving experience;

b compiling "deep" knowledge into more efficient "shallow" inference rules;

c machine learning.

2 Research needs to be pursued in the areas of:

a the representation of knowledge;

b inference methods;

c meta-level architecture for the construction of programs that can explicitly reason about and control their own problem-solving activity;

d user interfaces.

Progress in AI is impeded by an extreme shortage of manpower and equipment, both at universities and at laboratories that do AI research. The shortage of researchers is not likely to be alleviated for several years. A number of recommendations are made for dealing with these problems to increase the rate of research progress in the next ten years.

II. *Distributed Computing and Networking*

Fast-moving technology will result in the creation of multi-million-node computer networks by the end of the decade, accompanied by enormous opportunities for creative use. It will render obsolete much of what is now known about designing and controlling systems.

Among the areas of major importance are:

1 Theoretical foundations

2 Reliability

3 Privacy and security

4 Design tools and methodology

5 Distribution and sharing

6 Accessing resources and services

7 User environment

8 Distributed databases

9 Network research.

Resource problems are:

1 Level of funding

2 Manpower and education

3 Facilities

Specializing on the facilities issue, it has been amply demonstrated that hardware, software, research, and development are genuine experimental sciences, and that design ideas must be experimentally validated. There is a total lack of experience with larger (greater than 1,000-processor) systems and it is vital to have experimental testbeds on which algorithms can be validated. This is an especially fruitful area for industry-university collaboration.

III. *Highly Parallel Computing*

Parallel computing can overcome the basic physical constraints of uniprocessors, limitations which preclude the effective application of computing to problems of great economic importance and of national security concern.

Parallel computing is a *systems* issue. The state-of-the-art for parallel software and parallel formulation of significant

problems lags studies of parallel architecture. The issues of memory and I/O systems for parallel architectures lag the concepts of parallel computing engines.

The current bottleneck to progress is the difficulty of executing significant experimental studies which are essential to evaluation of total system concepts. To overcome this bottleneck, it is recommended that there be established:

1 Facilities for the development and evaluation of feasibility prototypes of systems where modeling has established design merit.

2 Facilities based on existing parallel processor systems for development and evaluation of algorithms and problem formulations.

3 Computer facilities to support simulation (or emulation) evaluation of detailed models of proposed architectures.

The recurring funding for there facilities and projects should, after a build-up period, reach $50,000,000 per year. A government-industry-university partnership is envisioned.

IV. *Information Science*

1 Provision of research support funds at a rate significantly in excess of the inflation rate. General areas in need of increased support are theoretical information science and behavioral aspects of information transfer programs.

2 Establishment of a CENTER FOR THE STUDY OF HIGHLY PARALLEL INFORMATION PROCESSING SYSTEMS that provides large storage capacities and high processing rates. The theoretical and experimental work envisioned for the Center would concentrate on the computer simulation of complex categorization algorithms of a highly parallel nature and interdisciplinary research to determine the critical elements of biological categorization algorithms to guide the machine experiments.

3 Promotion of the development of improved curricular materials for university majors and concentrations in information science and spurring the production of highly trained personnel.

4 Establishment of a KNOWLEDGE RESOURCES FACILITY, an experimental facility to aid researchers in explorations ranging from databases of text and images to knowledge bases that are beginning to play an important role in adaptive information processing strategies.

V. *Research in Support of Some Major Use Areas*

Emphasis is placed on the viewpoint of those who are attempting to apply computer science, computer engineering, and information science and additional pertinent supporting research.

A restricted group of use-areas was addressed, chosen so as to represent two quite different kinds of user communities: those likely to be tightly coupled to the computer science, computer engineering, and information science research communities, and those which are likely not to be so tightly coupled to them. The areas chosen, the nature of their coupling, and research recommendations are as follows:

1 *Office Automation* - likely to be tightly coupled to computer and information science.

Language and process development remains as a central issue; there are few effective ways for computer-naive professionals to express their domain-specific needs to the powerful engine in front of them. Applications languages and development tools are major research items.

2 *Pattern Recognition and Image Processing* - likely to be tightly coupled to computer engineering. Areas of research are:

a Highly parallel architectures to achieve higher speeds;

b Systematic and rapid mapping of architectural alternatives into VHSIC and VLSI hardware;

c The storage and retrieval of huge numbers of images in back-end databases;

d More efficient digital representations and languages for them;

e Exploration of alternatives to the digital representation and processing of images.

3 *Software Productivity* - likely to be tightly coupled to computer and information science.

There is a great need and potential for software productivity improvement. It is recommended that modest research projects focus on the coupling between software engineering research and its exploitation in software development environments.

4 *Large-Scale Systems, (including CAD/CAM, automated production, and civil engineering design aids)* - likely to be loosely coupled to all areas of computer and information science, and computer engineering research.

It is very much in the national interest to develop the knowledge base to design large-scale systems for:

a The construction, support, and coordination of selected large-scale systems used as test beds for research and shared by researchers;

b The coordination of funding by government and the private sector (including grants of equipment) in support of this research area; and,

c Aiding the achievement of overall research objectives by serving as a catalyst to the combined efforts of many investigators.

Relative to the problem areas above, it was concluded that there is a deep shortage of technically competent computer-literate engineers, scientists, and managers, aggravated by the poor state of equipment and facilities in universities relative to industry. Corrective measures are recommended.

VI. *Robotics*

The problem areas in robotics for which computer science and engineering appear to be best prepared to make

needed contributions are:

1 Languages for specifying robot tasks.

2 Planning of actions and motions.

3 Creating geometric models of complex objects and environments.

It is important that robot systems be flexible and easy to adapt to new applications. To this end:

1 Robot mechanisms must be universal, e.g., manipulators have six or more degrees of freedom to permit arbitrary positions/ orientations to be attained, even in the presence of fixed obstacles;

2 The programming system which drives a robot must be structured to allow robot software system builders, applications package writers, and maintenance and repair personnel to use it as conveniently as possible.

There need to be developed:

1 Better sensors and improved procedures for analyzing sensor-generated data (including multi-image vision systems);

2 Object recognition algorithms;

3 Environment-modeling software (including force-controlled motion primitive);

4 Computerized replanning techniques;

5 Robot locomotion.

The requirements of robotics are expected to make a significant reorganization of the present computer science and engineering curriculum necessary.

General Concerns

There were expressed several threads of concern by all of the Workshop groups, as follows:

1 A severe shortage of research manpower exists particularly in universities, in the fields embraced by the Workshop. Significant improvement is not expected during the next several years and the lack of a "critical mass" in some specialties portends undesirably slow research progress. A major government-industry-university effort is needed to deal with this problem.

2. There is still a shortage of science and engineering to serve as a base for applied research and development.

3 There is an increasing need for being able to customize rapidly a computer system with novel architectures for specific areas of research. This need is driven by applications involving words, as well as numbers. The counterpoint to this need is an increased ability to make efficacious use of complex computer systems that can now be constructed.

4 There is a paucity of large-scale computer systems in universities. The lack of access to such systems by university researchers is hindering research progress, both experimentally and theoretically, in all the topical areas which this Workshop addressed.

5 The level of research funding is far too low to support adequately the capabilities that exist and the maintenance of a national leadership position.

MEMBERS OF THE STEERING COMMITTEE

for the

INFORMATION TECHNOLOGY WORKSHOP

Robert F. Cotellessa, Chairman	Stevens Institute of Technology
Juris Hartmanis	Cornell University
Floyd H. Hollister	Texas Instruments, Inc.
Joel Moses	Massachusetts Institute of Technology
James H. Mulligan, Jr.	University of California, Irvine
Yoh-Han Pao	Case-Western Reserve University
Howard L. Resnikoff	Harvard University
Herbert Schorr	International Business Machine Corporation
Jacob T. Schwartz	New York University
Peter M. Will	Schlumberger Well Services
Martha E. Williams	University of Illinois, Urbana
Joe B. Wyatt	Vanderbilt University
Lotfi Zadeh	University of California, Berkeley

INFORMATION TECHNOLOGY WORKSHOP

January 5,6,7, 1983

DIRECTORY: COMMITTEE LISTING

ARTIFICIAL INTELLIGENCE

*Dr. David L. Waltz
Coordinated Science Laboratory
University of Illinois at
Urbana-Champaign
1101 West Springfield Avenue
Urbana, IL 61801
(217) 333-6071

Dr. Michael Genesereth
Computer Science Department
Stanford University
Stanford, CA 94305
(415) 497-0234

Dr. Peter Hart
Artificial Intelligence Laboratory
Fairchild Research & Development
Laboratory
4001 Maranda Avenue
Palo Alto, CA 94304
(415) 493-5375

Dr. Juris Hartmanis,
Walter R. Read Professor, Chairman
Department of Computer Science
Cornell University
405 Upson Hall
Ithaca, NY 14853
(607) 256-4052

Dr. Gary G. Hendrix
Symantec
306 Potrero Avenue
Sunnyvale, CA 94086
(408) 737-7949

Dr. Aravind K. Joshi
The Moore School of Electrical
Engineering - Rm 268
Department of Computer & Information
Science
School of Engineering and Applied
Science
University of Pennsylvania
Philadelphia, PA 19104
(215) 898-8540

Dr. John McDermott
Department of Computer Science
Carnegie-Mellon University
Schenley Park
Pittsburgh, PA 15213
(412) 578-2599

Dr. Thomas Mitchell
Department of Computer Science
Rutgers University - Busch Campus
Hill Center
New Brunswick, NJ 08903
(201) 932-3259

Dr. Joel Moses, Chairman
Department of Electrical & Computer
Engineering
Massachusetts Institute of Technology
77 Massachusetts Avenue
Cambridge, MA 02139
(617) 253-4601

Dr. Nils Nilsson
Artificial Intelligence Laboratories
SRI International
333 Ravenswood Avenue
Menlo Park, CA 94025
(415)859-2311

Dr. Robert Wilensky
Room 553/Evans Hall TE-01
Computer Science Division
Department of Electrical Engineering
and Computer Science
University of California
Berkeley, CA 94720
(415) 642-7034

Dr. William A. Woods
Principal Scientist
Bolt, Beranek & Newman
Artificial Intelligence
10 Moulton Street
Cambridge, MA 02238
(617) 497-3361

DISTRIBUTED COMPUTING AND NETWORKING

*Dr. Harold S. Stone
Department of Electrical and Computer
Engineering
The Commonwealth of Massachusetts
University of Massachusetts
Amherst, MA 01003
(413) 545-1971

Dr. David J. Farber
Department of Electrical Engineering
University of Delaware
Newark, DE 19711
(302) 738-2405

Dr. Robert Gallagher
Electrical Engineering Department
Massachusetts Institute of Technology
Cambridge, MA 02139
(617) 253-2333

Dr. Oscar N. Garcia, Chairman
Department of Computer Science and Engineering
Lib 630
University of South Florida
Tampa, FL 33620
(813) 974-4232

Dr. Hector Garcia-Molina
Electrical Engineering and
Computer Science Department
Princeton University
Princeton, NJ 08544
(609) 452-4633
Messages: (609) 452-4624

Dr. Lawrence H. Landweber
Computer Science Department
University of Wisconsin
Madison, WI 53706
(608) 262-1204

Dr. Edward D. Lazowska
Department of Computer Science - FR-35
University of Washington
Seattle, WA 98195
(206) 543-4755

Dr. Rick Rashid
Computer Science Department
Carnegie-Mellon University
Pittsburgh, PA 15213
(412) 578-2591, 2565

Dr. Robert Thomas
Bolt, Beranek & Newman
50 Moulton Street
Cambridge, MA 02238
(617) 491-1850

Dr. Andries van Dam
Computer Science Department
Brown University
Providence, RI 02912
(401) 863-3300

HIGHLY PARALLEL COMPUTING

*Dr. James C. Browne
Department of Computer Science
T.S. Painter Hall 3.28
University of Texas at Austin
Austin, TX 78712
(512) 471-5023

Dr. Bruce Arden
Department of Electrical Engineering
Princeton University
Princeton, NJ 08540
(609) 452-4640

Dr. Arvind
Massachusetts Institute of Technology
Laboratory for Computer Science
Room 535
545 Technology Square
Cambridge, MA 02139
(617) 253-6090

Dr. Forest Baskett
Digital Equipment Corporation
4410 El Camino Real
Los Altos, CA 94022
(415) 949-0770

Dr. William Buzbee
Los Alamos National Laboratories
P.O. Box 1663
C Division Office
Mail Stop 260
Los Alamos, NM 87545
(505) 667-1449

Dr. Mark Franklin
Department of Electrical Engineering
Washington University
St. Louis, MO 63130
(314) 889-6107

Dr. Robert M. Keller
3160 MEB
Department of Computer Science
University of Utah
Salt Lake City, UT 84112
(801) 581-5554

Dr. H.T. Kung
Carnegie-Mellon University
Schenley Park
Pittsburgh, PA 15213
(412) 578-2568

Dr. Duncan Lawrie
Department of Computer Science
University of Illinois
1304 West Springfield
Urbana, IL 61801
(217) 333-3373

Dr. Fred Ris
Senior Manager,
Computation-Intensive Systems
IBM Research
P.O. Box 218
Yorktown Heights, NY 10598
(914) 945-1973

Dr. Herbert Schorr
Vice President for Systems
I.B.M. Corporation
T.J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598
(914) 945-1285

Dr. Howard J. Siegel
Department of Electrical Engineering
Electrical Engineering Building
Purdue University
West Lafayette, IN 47907
(317) 494-3444

Dr. Lawrence Snyder
Department of Computer Science
Mathematics Building
Purdue University
West Lafayette, IN 47906
317) 494-6012

Dr. Robert Voigt
ICASE, Mail Stop 132C
NASA - Langley Research Center
Hampton, VA 23665
(804) 827-2513

INFORMATION SCIENCE

*Dr. Howard L. Resnikoff
Harvard University
100 Cotting House
Boston, MA 02163
(617) 495-1579

*Dr. Lotfi Zadeh
Department of Electrical Engineering
and Computer Science
University of California, Berkeley
Berkeley, CA 94720
(415) 642-4959

Dr. Richard Herrnstein
Department of Psychological Social
Relations
Wm James 730
Harvard University
Cambridge, MA 02138
(617) 495-3852

Dr. Michael Lesk
Room 2-C-465
Bell Laboratories
600 Mountain Avenue
Murray Hill, NJ 07974
(201) 582-6377

Dr. Roger Schank
Department of Computer Science
Yale University
P.O. Box 2158
New Haven, CT 06520
(203) 436-0606

Dr. Henry A. Sowizral
Computer Scientist
Information Science Department
The Rand Corporation
1700 Main Street
Santa Monica, CA 90406
(213) 393-0411

Dr. Joseph Traub, Chairman
Department of Computer Science
Columbia University
New York, NY 10027
(212) 280-1754

Dr. Donald Walker
Artificial Intelligence Center
SRI International
333 Ravenswood Avenue
Menlo Park, CA 94025
(415) 859-3071

Dr. Martha E. Williams
Professor of Information Science
College of Engineering
Coordinated Science Laboratory
University of Illinois at
Urbana-Champaign
1101 West Springfield Avenue
Urbana, IL 61801
(217) 333-8462

Mr. Joe B. Wyatt, Chancellor
Vanderbilt University
207 Kirkland Hall
Nashville, TN 37240
(615) 322-2168

RESEARCH IN SUPPORT OF SOME MAJOR USE AREAS

*Dr. Floyd Hollister
M.S. 238 08-220
Texas Instruments, Inc.
P.O. Box 226015
Dallas, TX 75266
(214) 995-0407

Dr. Gerald Estrin
Computer Science Department
BH 3732
University of California
Los Angeles, CA 90024
(213) 825-2786

Dr. K.S. Fu
School of Electrical Engineering
Purdue University
West Lafayette, IN 47907
(317) 494-4600

Dr. Robert J. Hocken, Chief
Division of Automated Production
Department of Commerce
National Bureau of Standards
CME, Building D-104
Washington, DC 20234
(301) 921-2577

Dr. W. Frank King
IBM Corporation
1000 Westchester Avenue
White Plains, NY 10604
(914) 696-6770, 6774

Dr. Daniel P. Loucks
Environmental Engineering
Hollister Hall
Cornell University
Ithaca, NY 14853
(607) 256-4896,7560

Dr. James H. Mulligan, Jr.
School of Engineering
University of California, Irvine
Irvine, CA 92664
(714) 833-6486

Dr. Philip N. Nanzetta, Deputy Chief
Division of Automated Production
Department of Commerce
National Bureau of Standards
CME, Building D-104
Washington, DC 20234
(301) 921-3474

Mr. Alfred M. Pietrasanta
IBM Systems Research Institute
205 East 42nd Street - 6th Fl
New York, NY 10017
(212) 983-7243

Dr. Robert J. Spinrad
Palo Alto Research Center
Xerox Corporation
3333 Coyote Hill Road
Palo Alto, CA 94304
(415) 494-4020

ROBOTICS

*Dr. Jacob T. Schwartz
Department of Mathematics & Computer Science
Courant Institute of Mathematical Sciences
New York University
Washington Square
New York, NY 10012
(212) 460-7209

Dr. Herbert Freeman
Department of Electrical, Computer & Systems Engineering
Rensselaer Polytechnic Institute
Troy, NY 12181
(518) 270-6330

Dr. John E. Hopcroft
Department of Computer Science
Cornell University
420 Upson Hall
Ithaca, NY 14853
(607) 256-4052

Mr. Stuart Miller
General Electric Company
Corporate Research & Development
Building 37 - Room 511
1 River Road
Schenectady, NY 12345
(518) 385-5865

Dr. Peter M. Will, Director
Product Systems
Schlumberger Well Services - Engineering
P.O. Box 4594
Houston, TX 77210-4594
(713) 928-4477

*Denotes committee chairman

AGENDA FOR

INFORMATION TECHNOLOGY WORKSHOP

January 5,6,7, 1983

Xerox International Center for
Training and Management Development
Leesburg, Virginia

5 January - Morning — Plenary Session
9:00 a.m. - 12:00 Noon

1 Overview - Dr. Jacob T. Schwartz, Courant Institute of Mathematical Sciences, New York University

2 Planning for research - Dr. Jack T. Sanderson, Assistant Director, National Science Foundation

3 Manpower and training - Dr. Kent K. Curtis, Head, Computer Science Section, National Science Foundation

4 Charge to subgroups - Dr. Robert F. Cotellessa, Provost, Stevens Institute of Technology

5 January - Afternoon: — Meetings of six subgroups
1:15 p.m. - 5:00 p.m. — Discussion based on respective foundation papers
Identification of subtopics to be addressed

5 January - Evening: — Dinner meeting of group chairmen (6:30 p.m.)

6 January - Morning: — Meetings of six subgroups
8:30 a.m. - 12:00 Noon
Enumeration of specific problem and opportunity topics
Related physical resource and manpower requirements

6 January - Afternoon: — Meetings of six subgroups
1:15 p.m. - 5:00 p.m.
Development of research agendas

Interface and institutional considerations

6 January - Evening: — Dinner meeting of group chairmen (6:30 p.m.)

7 January - Morning: — Meetings of six subgroups
8:30 a.m. - 11:30 a.m.
Generation and completion of research agendas

7 January - Afternoon: — Plenary reporting and critique session
12:45 p.m. - 2:45 p.m.

KEYNOTE PAPER
INFORMATION TECHNOLOGY WORKSHOP

RESEARCH IN COMPUTER SCIENCE: INFLUENCES, ACCOMPLISHMENTS, GOALS

Jacob T. Schwartz
Division of Computer Science
Courant Institute of Mathematical Sciences
New York University

TABLE OF CONTENTS

1.0 *TILL NOW: TECHNOLOGY AS THE DRIVING FORCE.*

My charge in this talk is to lay out a perspective concerning research in computer science; I must of course warn you all that any such perspective is inevitably personal.

Let me begin with the remark, taken from the recent NRC Study 'Roles of Industry and University in Computer Research and Development'1, that two basic ideas, namely Babbage's stored program concept and its triumphant transistor/ microcircuit implementation, continue after thirty years to drive the computer field as a whole. These ideas have proved so rich in consequence as to ensure the practical success of our field, independent of any other accomplishment. Since, as Turing emphasized, the Babbage/VonNeumann processor is computationally universal, any computational paradigm is accessible to it, so that it can be improved in speed and size only, never in fundamental capability. Nevertheless, the successes of microcircuit technology have enabled computer speed and size to grow in an astounding way, confronting workers in the field with a continually widening range of opportunities. For example, discs have led to databases, communication technology has led to ARPANET, and microchips to interest in distributed computing.

Based upon this technological/economic progress, the applications of computer technology have broadened rapidly, much more rapidly than the technology itself has deepened. Though constrained by software approaches that are barely acceptable, an extraordinary dissemination of computer technology has begun and will surely continue through the next decade.

The computer has become a household item, and besides its business applications, now stands at the center of a large entertainment industry. This guarantees the use of mass production techniques, implying yet further cost reductions. We can expect that a lush computational environment, represented by families of new personal computers whose performance will rapidly rise to several MIPS, will become commonplace for the scientist and business executive over the next few years, and with some additional delay will become common in homes. Continued improvements in magnetic recording densities, plus use of the extraordinary storage capacity of videodisk, will make large amounts of fixed and variable data available in the personal computer environment which we are beginning to see. Where advantageous, the computational power of workhorse microprocessor chips will be further buttressed by the use of specialized chips performing particular functions at extreme speeds. Chips for specialized graphic display and audio output functions are already available, and other such chips, performing signal analysis, data communication, data retrieval, and many other functions will be forthcoming. Finally, cheap processor chips make new generations of enormously powerful, highly parallel machines economically feasible.

The next few years will begin to reveal the fruits of these new opportunities. I believe, for example, that we will see a flowering of computer applications to education which will reflect our new ability to integrate microcomputers, computer graphics and text display, audio and video output, large videodisk databases, including stills and moving images, and touch-sensitive screens into truly wonderful, game-like learning environments. The video-game industry is surely alive to these possibilities, and I believe that its activity will affect both education and publishing more deeply than is generally suspected. All this will be part of the development of a world of personal and business information services more comfortable and attractive than anything we have yet seen.

2.0 *TILL NOW: THE SWAMP OF COMPLEXITY.*

Thus, hardware trends, reflecting the microcircuit engineer's astonishing progress, face us with the opportunity and requirement to construct systems of extraordinary functionality. However, even in the best of cases, this increased functionality will imply increased systems complexity. Moreover, since lack of firm understanding of how to proceed with software has never fully restrained the

1 Report of the Basic and Applied Research Panel, Computer Science and Technology Board, National Research Council, November 1982.

drive to increased functionality, huge, sprawling, internally disorganized and externally bewildering systems of overwhelming complexity have already been built, and have come to plague the computer industry. These dimly illuminated, leaky vessels force more of their crew to bail, and to grope about for half-hidden control switches, than are available to row forward. The situation is ameliorated in the short term, but at length exacerbated, by the economic 'softness' of software: costs are incurred incrementally rather than in a concentrated way; no single detail is difficult in itself, but the accumulation of hundreds or thousands of discordant details makes the overall situation impossible. This is the 'software crisis'.

Language and software systems designers have attempted to find ways of slowing the growth of this sprawling disorganization, and have succeeded to at least some modest degree. First assemblers, then compilers, and then compilers for progressively higher levels of language have been developed to allow the pettiest and most onerous levels of detail to be handled automatically. This has helped overcome one of the major obstacles to stabilization and re-use of software, namely pervasive infection of codes by machine related details. This is part of the reason why the first generation of symbolic programming languages, specifically FORTRAN and COBOL, proved such a boon to the industry. Their success is shown by the large libraries of numerical, statistical, engineering, and other widely used applications programs that have accumulated in them. LISP has performed something of the same good service for workers in artificial intelligence. Substantially higher levels of language, which enable the programmer to use comfortable, human-like levels of abstraction somewhat more directly, also promise to make it easier to move major pieces of software from one application to another. It is fortunate that falling hardware costs are encouraging the use of these computationally expensive languages.

During the last few years, the technology of software transportability, exemplified by UNIX, various PASCAL compilers, and numerous microcomputer software systems has become well established. Emphasis on ideas of modularity and information hiding, about to receive a major test in the new DoD ADA language, should also contribute to the re-usability of software modules.

Better support tools have been provided to aid the programmer's work, and this improvement is continuing. Today's fast-turn-around environment, with its interactive editing, text-search facilities, and quality displays is a substantial boon. However, much more remains to be done to integrate the programmer's editing/compiling/debugging/tracing/version-control work environment into a truly helpful support system. This is a sprawling, complex problem, and I will not even begin the enumeration of its many facets. Let me note, however, that experimental development of more sophisticated program analysis tools, e.g. routines which pinpoint bugs by searching for textual incongruities in programs, is going forward and should lead to useful products.

Conceptual progress in the systems area has been linked to the discovery of families of objects, operations, and notations that work well together in particular application areas. Some of these object/operator families have proved to be of relatively general applicability, others to be powerful, but only in limited domains. Important examples are: the use of BNF grammars, variously annotated and 'attributed', for the syntactic and semantic description of languages has been central to the development of 'compiler-compilers'. Tony Hoare's notion of 'monitor' furnishes a significant conceptual tool for organizing the otherwise confusing internal structure of operating systems. The well-chosen family of string-matching operations of the SNOBOL language, with the success/failure driven, automatically backtracking control environment in which they are embedded, allows many string-related applications to be programmed with remarkable elegance and efficiency.

3.0 *EMERGENCE OF AN INTELLECTUAL PRECIPITATE HAVING LONG-TERM SIGNIFICANCE.*

Though buffeted, and sometimes rendered obsolete, by steadily changing technology, three decades of work by language and systems designers have given us products whose broad use testifies to a substantial degree of usability. However, since in this area it is so hard to distinguish real technical content from the arbitrary effects of corporate investment, salesmanship, and accidents of priority, it is gratifying that a solid theoretical inheritance has begun to emerge from this raw systems material. At the very least, this gives us an area in which the individualistic assertions of taste and the irresolvable debates characterizing the systems area yield to clear criteria of progress and theoretically based understanding of limits and tradeoffs. Whereas systems will change with technology, this work, like the mathematics it resembles, will be as important a century from now as it is today.

Central to this, the intellectual heart of computer science, stands the activity of algorithm design. By now, thousands of algorithmic clevernesses have been found and published. Of all the work that could be adduced, let me draw attention to four strands only. Many very useful and ingenious data structures, supporting important combinations of operations with remarkable efficiency, have been found. Hash tables, B-trees and the many variants of them that have been analyzed in the recent literature, and the specialized representations used to handle equivalence relationships efficiently are some of the prettiest discoveries in this subarea. Ingenious ways of arranging and ordering work so as to minimize it have been uncovered and exploited. The well-known divide-and-conquer paradigm, and also Bob Tarjan's many depth-first-spanning-tree based graph algorithms, exemplify this remark. Clever combinational reformulations have been found to permit efficient calculation of graph-related quantities which at first sight would appear to be very expensive computationally. The fast algorithms available for calculation of maximal matches in constrained-choice problems, maximum network flows, and also some of Tarjan's algorithms for solving path problems in graphs all illustrate this third point. Finally, deep-lying algebraic identities have been used to speed up important numerical and algebraic calculations, notably including the discrete Fourier transform and various matrix operations.

The techniques of formal algorithm performance analysis, and the fundamental notion of asymptotic performance of an algorithm, both put squarely on the intellectual map by Donald Knuth, give us a firm basis for understanding the significance of the algorithms listed in the preceding paragraph and the thousands of others that have appeared in the computer science literature. An algorithm that improves the best known asymptotic performance for a

particular problem is as indisputably a discovery as a new mathematical theorem. This formal criterion of success, by now firmly established, has allowed the work of algorithm designers to go forward in a manner as systematic and free of dispute as the work of mathematicians, to which it is similar.

Studies of the ultimate limits of algorithm performance have deepened the work of the algorithm designer, by putting this work into a mathematical context drawing upon and deepening profound ideas taken from mathematical logic and abstract algebra. Landmarks here are the Cook-Karp notion of NP-completeness, which has proved to draw a very useful line between the feasible and infeasible in many algorithmic areas close to practice; the work on exponentially difficult problems deriving from Meyer, Stockmeyer, and Rabin; and various more recent investigations, by 'pebbling' and other ingenious combinatorial methods, of time-space tradeoffs in computation.

The mathematical affinities of most of this work are with mathematical logic, combinatorics, and abstract algebra, which till now have been the branches of mathematics of most concern to the computer scientist. However, developments now afoot seem likely to renew the ties to classical applied mathematics which were so important in the earliest days of computing.

4.0 *TRENDS.*

Some of the dominant research currents of the next decade will merely deepen what is already important, others will bring matters now attracting relatively limited attention to center stage. The VLSI revolution will surely continue, dropping the price of processing and storage. As already stated, this ensures that most of the existing army of programmers will be occupied in keeping up with the developmental opportunities that this technology will continue to provide. To avoid complete paralysis through endless rework of existing software for new hardware, software approaches which stress transportability and carry-forward of major systems, plus re-usability of significant software submodules, must and will become more firmly established. Growing understanding of the most commonly required software subfunctions should allow many of them to be embodied in broadly usable software, firmware, or hardware packages. Besides numerical, statistical, and engineering packages of this sort, operating system, database, communication, graphics, and screen management packages should all emerge.

The steadily broadening applications of computers will draw algorithm development into new areas. Computational geometry, i.e. the design of efficient ways for performing many sorts of geometric computations, has flourished greatly since its first buddings a few years ago and gives a good indication of what to expect. This area of algorithmics contributes to VLSI design, CAD/ CAM development, and robotics, all of which have nourished it.

Robotics can be expected to play a particularly large role in bringing new directions of investigation to the attention of algorithm designers. Though it may be expected to draw upon many of the other branches of computer science, robotics will have a different flavor than any of them, because in robotics, computer science must go beyond the numeric combinatorial, and symbolic manipulations that have been its main concern till now: it must confront the geometric, dynamic, and physical realities of three-dimensional space. This confrontation can be expected to generate a great deal of new science, and will greatly deepen the connections of computer science to classical applied mathematics and physics.

If, as I expect, robotics becomes as central to the next few decades of computer science research as language, compiler, and system-related investigations have been during the past two decades, significant educational and curricular problems will confront many computer science departments. Computer science students' knowledge of the basics of applied mathematics and physics is small at present, and I suspect that it is diminishing as the systems curriculum expands and older connections with mathematics are lost. For roboticists, this trend will have to be reversed: solid training in calculus and differential equations, linear algebra, numerical analysis, basic physics, mechanics, and perhaps even control theory will be necessary. Students going on to graduate work and research careers will also have to concern themselves with computational algebra, computational geometry, theories of elasticity and friction, sensor physics, materials science, and manufacturing technology.

Beyond the educational problem of how to squeeze all of this material into an already crowded curriculum, administrative problems of concern to sponsoring agencies will also arise. Presently, computer acquisition is regarded as a capital investment: once acquired, processors, storage units, terminals, printers are expected to remain in productive service over an extended period and to be used by many researchers for many purposes. Though some robotic equipment will have this same character, other equally important (and expensive) equipment items will, like the equipment of the experimental physicist, be useful only for specific, limited sequences of experiments. Robotics researchers will therefore have to learn how to define experiments likely to produce results worth high concentrated costs, and funding agencies will have to learn how to pay bills of a kind new to computer science.

One last area of practical and theoretical development deserves mention. Large-scale parallel computers seem certain to appear during the next few years as super-performance supplements to the ubiquitous microchip. Many university groups are currently preparing designs of this sort, and industry, not excluding Japanese industry, is beginning to become interested. This appears certain to make parallel algorithm design and the computational complexity theory of such algorithms active research areas during the next few years.

5.0 *ELDORADO.*

Though presently revolving around technology and such prosaic accomplishments as putting a word-processor in every office and home, the computer development leads forward to something immensely significant: the development of artificial intelligences which can at first match, and perhaps soon thereafter greatly surpass, human intelligence itself. The belief that this is possible motivates many computer scientists, whether or not they concern themselves directly with this subfield of computer science. It would be hard to overestimate the consequences of such an outcome.

Since this is the deepest perspective of our field, I shall conclude this talk by addressing it briefly. Concerning artificial intelligence, I think it well to comment with high

hope concerning its long-term future, skeptically concerning its accomplishments to date, and in repudiation of the relentless overselling which has surrounded it, which I believe discourages those sober scientific discernments which are essential to the real progress of the subject.

Let me begin by subscribing to the materialist, reductionist philosophy characteristic of most workers in this field. I join them in believing that intelligent machines are indeed possible, since like them I view the human brain as a digital information processing engine, a 'meat machine' in Marvin Minsky's aptly coarse phrase, wonderfully complex and subtle though the brain may be. This said, however, we must still come to grips with the main problem and dilemma of the field, which can, I believe, be formulated roughly as follows. To communicate with a computer, one must at present *program*, i.e. supply highly structured masses of commands and data. Communication with other persons is much easier, since they can digest relatively unstructured information, and can supply the still elusive steps of error correction and integration needed to use such information successfully. Thus only to the extent that a computer can absorb fragmented material and organize it into useful patterns can we properly speak of it as having 'intelligence'.

Researchers in artificial intelligence have therefore sought general principles which, supplied as part of a computer's initial endowment, would permit a substantial degree of self-organization thereafter. Various formal devices allowing useful structures to be distilled from masses of disjointed information have been considered as candidates. These include graph search, deduction from predicate algorithms, and, in today's misnamed 'expert' systems, the use of control structures driven by the progressive evaluation of a multiparameter functions whose parameters are initially unknown.

The hoped-for, and in part real, advantages of each of these formal approaches have triggered successive waves of enthusiasm and each approach has lent particular terminological color to at least one period of work in artificial intelligence. Any collection of transformations acting on a common family of states defines a graph, and discovery of a sequence of transformations which carries from a given to a desired state can therefore be viewed as a problem in graph search. Accordingly, transformation of graph into path can be regarded as a general formal mechanism which permits one to derive something structured, to wit a path, from something unstructured, namely its underlying graph. Hope that this principle would be universally effective characterized the first period of work in artificial intelligence, embodied in systems such as the 'General Problem Solver', and in languages such as Planner.

The relationship of a proof in predicate calculus to the order-free collection of axioms on which this proof is based gives a second principle allowing ordered structures (i.e. proofs) to arise from unordered items of information (i.e. axioms). Because of its deep connections to mathematical logic and its great generality, this formal approach seems particularly promising, and in its turn encouraged over a decade of work on resolution proof methods and their application. Today a new approach stands at the center of attention. Though considerably less deep and general than either the graph search or the predicate techniques which preceeded it, this ask-and-evaluate sequencing paradigm, much used in today's 'expert' consultant systems, has found enthusiastic advocates.

Sustained and patient work aimed at full elucidation of the real, albeit very partial, insights inherent in each of these approaches is certainly appropriate. All, however, face a common problem: their efficiency decreases, sometimes at a catastrophic rate, as they are required to deal with more information and as the situations which they try to handle are made more realistic. It is well-known, for example, that computerized predicate logic systems which can easily derive a theorem from twelve carefully chosen axioms will often begin to flounder badly if three more axioms are added to these twelve, and will generally fail completely if its initial endowment of axioms is raised to several dozen. Similarly, graph search methods that work beautifully for monkey-and-bananas tests involving a few dozen or hundred nodes fail completely when applied to the immense graphs needed to represent serious combinatorial or symbolic problems. For this reason I believe it very premature to speak of 'knowledge-based' systems, since the performance of all available systems degenerates if the mass of information presented to them is enlarged without careful prestructuring. That this is so should come as no surprise, since, not knowing how to specialize appropriately, we deal here with mechanisms general enough to be subject to theoretically derived assertions concerning exponentially rising minimal algorithmic cost. This observation reminds us that work in artificial intelligence must learn to reckon more adequately with the presence of this absolute barrier, which stubborn over-optimism has till now preferred to ignore.

Then along what lines is progres toward the goal of artificial intelligence to be expected? Through incremental, catch-as-catch-can improvement of our capacity to mimic individual components of intelligent function, such as visual pattern recognition, speech decoding, motion control, etc? By integrating algorithmic methods into systems which use the traditional A.I. techniques only as a last resort? Through eventual discovery of the A.I. philosopher's stone, namely some principle of self organization which is at once effective and efficient in all significant cases? By applying vastly increased computing power? We do not know, but the attempt to find out will continue to constitute the most profound, though not necessarily the most immediate, research challenge to computer science over the coming decades.

COMPUTER MANPOWER - IS THERE A CRISIS?

Kent K. Curtis

National Science Foundation
Washington, D.C., 20550

ABSTRACT

Factors influencing the supply and demand of computer manpower are analyzed in the context of available data on scientific manpower including statistics on degrees awarded in various disciplines at the Bachelor's, Master's, and Ph.D. levels, faculty mobility, job mobility among professionals, starting salary trends, comparative unemployment statistics and economic projections. It is found that there is a shortage of computer manpower which is expected to persist for the foreseeable future but that society is responding, perhaps as rapidly as possible, to provide the trained people required by business, industry and government. Only the educational institutions of the U.S. have what might be described as a crisis, a staffing problem which seems to have no solution within the context of normal supply/demand forces.

TABLE OF CONTENTS

1.0 *INTRODUCTION*

Although the subject of computer manpower supply and demand is of intense interest to many people, there are no experts. This is surprising because the gathering of statistics on all kinds of subjects is one of our national pastimes; yet, in this area the statistics we have in great abundance do not lead to any clear conclusions. Why?

For one thing, the question is ill-defined. There is no consensus on who should be included in the manpower pool being studied. The boundaries of every occupation are fuzzy, of course, but in the computer related occupations we don't even know where the center of gravity lies. The Bureau of Labor Statistics counts people according to occupational classifications which are approximately the same as the Federal government's job classifications. These include titles such as computer specialist, systems analyst, programmer, computer technician, etc., each including a wide range of educational levels, and the size of each group is so large (hundreds of thousands) the academic research community cannot be identified in any of them. The BLS projections are interesting, provocative, and display useful trends but they are not helpful for analyzing the problems of computer science faculties in universities.

John Hamblen, who has made several surveys of computer science-related programs in educational institutions of the United States[1] has provided a very valuable service, especially for the supply side of the question. However, his surveys do not include the many people who go into computer related jobs after graduating in physics, mathematics, economics, or many other disciplines. His summary statistics also need careful interpretation when they are applied to the problem of university faculties because he includes programs in Data Processing, Information Processing, and other titles, which expand his population of academic programs beyond what most computer scientists and computer engineers acknowledge as their domain of interest.

The National Science Foundation, which has a congressional mandate to monitor the supply and demand of engineers and scientists in the United States, only recently began to count computer scientists as a separate group. Until then they had only one entry called computing theory as a sub-category of mathematics. Even now, in classifying academic departments and gathering departmental statistics, they label all departments which have compound names, e.g. Electrical Engineering and Computer Science, by the first adjective. Thus, the departments at MIT or Berkeley make no contribution to NSF's statistics on computer science departments because they appear only in the numbers for Electrical Engineering.

On the other hand, the annual survey of departments done by Professors Samuel Conte of Purdue and Orin Taulbee of the University of Pittsburgh[2] restricts its domain to Ph.D.-granting departments which carry the label of computer science. As a result, although it includes some departments which are overlooked by the NSF survey, it omits all of those which are labeled computer engineering or are computer science options in departments of mathematics or business administration. In brief, all of the available data needs a great deal of interpretation, interpolation or extrapolation to make it useful for a study of manpower dynamics. This leaves every analysis open to question.

Another reason for confusion is that the dynamics of the computer manpower supply and demand are without precedent. We are experiencing an almost continuous change in the all-pervasive technology of information processing which creates a sudden, almost infinite demand for specialized talent. The industrial revolution which caused such tremendous social upheavals in the western world proceeded at a snail's pace by comparison. As a result,

history offers very limited guidance in the analysis of the present situation and gives no help whatsoever in predicting the immense transients that may be induced by such a strong forcing function as the computer revolution.

Nevertheless, laying all of these caveats aside, some interesting observations can be made which offer food for thought. In this paper, some data will be shown which exhibit convincing trends about supply, although not reliable absolute magnitudes about either supply or demand, and some conclusions will be drawn which offer a plausible framework for understanding the data and explaining the phenomena we are observing. These observations and interpretations have been exposed to several industrial managers and academic administrators who have found them to be consistent with their experience.

2.0 *IS THERE A CRISIS?*

What evidence do we have that there is a shortage of computer professionals? The most prominent is the anecdotal evidence which abounds. Everyone has heard many of the same kind of stories, and they need not be repeated here, but for the record let me show some data which support some of the most common.

o *Students are not entering graduate school* but are being lured by attractive salaries and professional opportunities at the bachelor's level. Note the ratios in Fig. 1. Computer Science has the lowest ratio of graduate degrees to bachelor's degrees of any discipline by a wide margin. There is no doubt that the bachelor's degree is proving to be a sufficient entree into the professional computer field for most people.

o *Graduate students are leaving graduate school* without completing their Ph.D.'s. In this case, note the ratios in Fig. 2. Either that statement is true or an inordinate number of people are entering graduate school only for the master's degree. As Fig. 3 shows, there is a higher ratio of part-time students to full-time students in Computer Science than in any other discipline. Since most part-time students are probably not Ph.D. candidates or planning to go on for a Ph.D., we can surmise that there are, indeed, more terminal master's students in Computer Science than in other disciplines. Even after subtracting out that difference, however, the number of Ph.D.s in Computer Science remains singularly low, indicating that many prospective Ph.D. students are dropping out. Anecdotal evidence strongly supports that conclusion.

o *Faculty are leaving academia for industry.*[3] The Computer Science Section of NSF conducted a faculty mobility study to understand better the reasons faculty change jobs. The number of faculty in Computer Science who made a job change in 1980, the year of the survey, was also estimated. Some results are shown in Fig. 4. The survey confirmed that a substantial fraction of the faculty changing jobs is moving to industry. Other data shows that the percentage going into industry is higher in the computer field than in any other discipline, even other branches of engineering. Fig. 5 shows the percentage of faculty moving into industry for several areas of engineering. It is clear that the loss of faculty to industry is twice as high in computer science as in any other field of engineering. Finally, Fig. 6 shows that the computer science faculties are not being replenished by new Ph.D.'s. Few Ph.D.s are being produced and only slightly more than half of those go into academic careers. This is unusually low in comparison with other disciplines.

It is interesting at this point to ask what are the factors influencing the career choices made by faculty and students when they are considering whether to enter industrial or academic careers? Fig. 7 shows the factors uncovered by the NSF survey of faculty mobility. It is noteworthy that salary was not the most compelling reason. Instead, "institutional disincentives" were. These include, of course, the uncertainty of tenure, a problem unique to academic institutions, which creates a feeling of job insecurity and frustration among young people who are starting academic careers. Other types of "institutional disincentives" cited most often were the difficulty and hassle associated with getting travel funds for professional conferences or other types of support for leading normal professional lives, heavy teaching loads because of the great influx of students into computer courses, and inadequate support of research, especially research in computer software or hardware systems. Few university administrators have experience with that kind of research which requires a high degree of support for facilities, time, and effort but is slow in yielding papers which can be published in the normal scientific literature. Universities have difficulty evaluating such work and this influences allocation of scarce university research funds or considerations of promotion. Many young computer researchers perceive industry as more hospitable and more rewarding for that kind of research.

Other anecdotal evidence of a shortage of computer manpower includes stories of intensive but often frustrating recruiting by industrial and academic organizations (just look at the ads in any journal or newspaper) and the fantastic salaries being offered. Many students, especially those with some relevant experience, have received amazing offers and some people have doubled their salaries by changing jobs. Statistical evidence to back up these stories is also available. Fig. 8 shows the increase in starting salaries for baccalaureate degree graduates during the period from 1974 - 1979. Starting salaries are used for this comparison because they are more flexible than salaries for established professionals and tend to be more sensitive as a barometer of hiring pressure. Fig. 9 shows that unemployment has been singularly low for computer professionals. Fig. 10 shows that computer science has had a higher growth rate in employment than any other field. Fig. 11 brings this data up to date by showing that only computer science had increased employment opportunities in 1982. Finally, a recent study by NSF, Fig. 12, shows that this imbalance in supply and demand is expected to persist for computer-related professions for the foreseeable future, even in the face of pessimistic economic assumptions. In fact, only the computer profession and aeronautical engineering show strongly encouraging prospects even under assumptions for the future of the economy which are very favorable to high technology.

Finally, let me note a rather general worry concerning education in the United States. Fig. 13 shows a suggestive correlation between trends in world trade and trends in technical education. One must be careful not to oversimplify, but this table is provocative. Something is out of balance which will redound to the long term disadvantage of our country.

3.0 *HOW DOES THE NATION RESPOND?*

Granted that there is an imbalance between supply and demand for computer professionals which is likely to persist for many years, how does society respond? We see part of the response quite clearly on our college campuses. Fig. 14 shows the total number of degrees awarded at the bachelor's, master's and Ph.D. levels for the last several years. There has been almost no change. A comparison with the demographic data shown in Fig. 15 shows recent degree data to be consistent with demographic data. Assuming that consistency continues, we should expect a decrease of 20% to 25% over the next 15 years in the annual production of degrees. However, let us look at some of the fine structure. Figures 16 to 37 show data on degrees awarded in each of several fields of science, engineering, the arts, education, and the humanities for the last several years. Obviously there are large migrations in student interest toward engineering and away from education, letters and the humanities. Now look at Fig. 38 for computer science. That is true exponential growth, a more rapid increase in bachelor's degrees than in any other discipline. Undergraduates are swarming to major in computer science. Master's degrees are also up by almost 60%. These changes are imposing immense stresses on the institutions of higher education, causing large imbalances in teaching loads, a huge shortfall in capital funds for instructional and research equipment and a mad scramble for faculty slots and persons to fill them as they become available. Many institutions are being forced to limit enrollments in computer science, an action which is difficult to accept in either private or public institutions because of the traditional assumptions of our country concerning universal educational opportunity. Indeed, in this discipline we are being propelled toward a system of limited educational privilege, comparable to the "numerus clausus" of Germany, for example, which carries with it the potential for long-term problems, already evident in Germany, of unemployment or under-employment among educable youth due to lack of opportunity for appropriate education.

It must be noted at this point that there have been rapid increases in demand for education in some other fields during the 1950's and 1960's. Note, for example, bachelor's degrees in some social sciences, biology and business. However, total college enrollments and federal support for education were increasing very rapidly during those periods, also. As a result, enrollments in all fields were increasing and the changes were primarily differential rates of increase into some fields. These were readily absorbed by the quite elastic, expanding university resources which existed at that time. Now, colleges and universities have static, soon to be declining, enrollments, and limited, inelastic resources, but are confronted with shifts of student interest of unprecedented proportions. It is no wonder that our educational institutions are feeling stress.

Setting these institutional problems aside for a moment, however, we see that one of the natural responses of our society to the shortage of computer professionals is to expand the number of students being educated for the computer professions as rapidly as our educational system can adjust. This is happening as a result of natural selection by students of preferred courses of study and is having a significant impact on the supply side of the balance.

Another indicator of social response is shown in Fig. 39. This shows, for 1978 bachelor's degree or master's degree recipients, the ratio of people employed in each field in 1980 to those who received a degree in that field in 1978. For example, chemistry shows a slight gain at the bachelor's level, but none at the master's. Engineering comes out even at the bachelor's level but has a marked decrease at the master's level. The computer professions, on the other hand, are sweeping people up from all disciplines like a giant vacuum cleaner, with people being retrained on the job or through part-time education as necessary, but moving to fill the void.

These effects are just what one should expect in a free market. Demand is creating supply by utilizing the job mobility of young professional people. Such job mobility has surely been facilitated by the computer experience which many college students now get as part of their normal undergraduate education regardless of their major subject, and NSF's program during the sixties to help colleges and universities acquire computing facilities for research and education did much to make that possible. It will be even further increased as personal computers become integrated into the educational environment of every student. These are natural economic and social processes which are progressing rapidly, and it is not clear what government can do, if anything, to facilitate them. It is true that capital funds for universities and colleges to acquire equipment are one bottleneck, but we hope that may be relieved by recent changes in tax laws which are intended to encourage donations of equipment to academic institutions. Obtaining qualified faculty is a greater and more imponderable problem, but faculty are being retrained and faculty slots are being reallocated. Indeed, the academic environment is changing, perhaps as fast as the inertial factors of tenure and the highly specialized nature of academic jobs allow, so that it is not clear that government action could have much leverage. Perhaps steps to increase computer literacy at the junior or senior high-school level would help relieve the service-course teaching burden in colleges and universities, but that movement, too, is already in progress at a rapid rate through natural processes which are subject to their own social inertia.

In brief, the natural working of free-market conditions for computer manpower supply and demand is leading to an increase in supply which is suitable for most purposes. The supply is not increasing as rapidly as it could be absorbed but perhaps as fast as society can respond, and there is nothing properly termed a computer manpower crisis in business, industry, or government. Acceptable people are becoming available in steadily increasing numbers and the only real effect of the shortage is a somewhat inflated price for labor, a differential which provides the fuel that powers social adjustment.

The situation in the academic world is quite different. Academic jobs are so specialized and require such long periods of education and training that job mobility is very low[4] and each class of academic institution can draw upon only one source of supply for labor. Elementary and secondary schools can use only people with proper education credentials to satisfy state and local certification standards. As the statistics on degrees in education show, that is a rapidly diminishing pool. Institutions of higher education, on the other hand, are restricted to the pool of Ph.D.s

for their labor supply and in computer science that is a small supply indeed, far out of balance with the expanding need. We must conclude that the educational institutions of the country cannot obtain the labor they need and have poor prospects of finding it in the near future. They face a real crisis. The migration of student interest is working against them, not in their favor, and the job mobility which allows so many people to enter computer professions in business, industry and government, is not effective for educational institutions because of their highly specialized job requirements. Let us analyze their dilemma somewhat further in the next section.

4.0 *THE CRISIS IN COMPUTER EDUCATION*

4.1 *Higher Education*

Let us consider the conundrum facing the computer field in higher education first. It is experiencing an exponentially increasing demand for its product with an inelastic labor supply. How has it reacted? NSF has made a survey of the responses of engineering departments, including computer science departments in schools of engineering, to the increasing demand for undergraduate education in engineering. There is a consistent pattern in their responses and the results can be applied without exception to the computer field whether the departments are located in engineering schools or elsewhere. 80% of the universities are responding by increasing teaching loads, 50% by decreasing course offerings and concentrating their available faculty on larger but fewer courses, and 66% are using more graduate-student teaching assistants or part-time faculty. 35% report reduced research opportunities for faculty as a result. In brief, they are using a combination of rational management measures to adjust as well as they can to the severe manpower constraints under which they must operate. However, these measures make the universities' environments less attractive for employment and are exactly counter-productive to their need to maintain and expand their labor supply. They are also counter-productive to producing more new faculty since the image graduate students get of academic careers is one of harrassment, frustration and too few rewards. The universities are truly being choked by demand for their own product and have a formidable people-flow problem, analogous to but much more difficult to address than the cash-flow problem which often afflicts rapidly growing businesses. There are no manpower banks which can provide credit.

What flexibility does our society have to cope with this dilemma? Retraining existing faculty can be done only to a small degree. Faculty slots are being reallocated as they become available but that is a slow process in a period of inflexible or declining budget resources. Even so, universities cannot fill all of the slots they have available. The net result is that the lack of lateral job mobility among faculty from one discipline to another coupled with the small supply of the only type of labor universities and colleges can use forces them to become less efficient and effective producers when confronted with strong shifts in market demand for degrees. They become burdened with people whose skills are not in demand, which increases their non-productive costs, at the same time they overwork people whose skills are in short supply, which threatens the quality of their educational product. Some people have begun to doubt whether universities can play, in computer science, their traditional role as the supplier of trained professional manpower. As a consequence, there are moves to establish specialized technical schools such as the Wang Institute or to rely more heavily on on-the-job training. If one could argue persuasively that this is a temporary or transient phenomenon, one might make a compelling case for federal government intervention to buffer the non-productive extra costs of institutions while they adjust to new conditions. In the absence of a sufficient pool of qualified persons to hire as faculty, however, it is difficult to make that case.

On the other hand, if universities and colleges were made more attractive places of employment, they could compete more effectively for the available talent and motivate more students to prepare for and choose academic careers. It is true that this question of attractiveness is a relative one which is influenced by economic conditions. Poor economic conditions in business and industry make universities relatively more attractive but the problem in the computer field is not relative, it is one of absolute numbers and we need a long-term, stable solution. Our NSF survey has indicated the factors to be addressed in any attempt to accomplish this. The "institutional disincentives" are most important and can be modified only by the institutions themselves--either more manpower or else limitations on enrollments to reduce work loads; more institutional financial resources allocated for staff travel, secretarial support and other normal professional needs to reduce harrassment and improve morale; more effective, or at least more apparent, consistency between the work required of faculty, namely, teaching, research and participation in committees, and the criteria used for promotion to increase the sense among young faculty that they work in a rational system with reasonable job security. Of these, the availablity of manpower is the factor which seems to be beyond institutional control the most. Yet if the institutions cannot control this factor, neither can anyone else, so at least we must examine it from the perspective of the universities. Clearly, the objective should be to either expand the manpower pool which universities may draw upon or else reduce their manpower requirements. Many universities have already expanded their pool in one way by turning to temporary or adjunct staff to help with the teaching load. If the manpower shortage is a transient problem, this is surely the best way to deal with it. It is arduous--one university I visited recently is recruiting and managing 35 to 40 temporary teaching staff each year drawing upon local industry--and creates severe problems of quality control, but the educational enterprise can probably be continued in this mode for some time if there is hope of improvement. From the data at hand, however, it is not clear how equilibrium between supply and demand of permanent staff can be realized in less than a generation which is a long time.

Another possibility would be to reduce manpower requirements by using technology both to make education more efficient and to move some aspects of computer training out of the universities into pre-college curricula. This approach has limited promise. It would clearly help reduce the demand for service courses in computer literacy and programming, but the hard-core dilemma concerns majors in computer science, not the broad community of users. We know very little, now, about applying technology to make education in advanced courses more efficient and it may take at least a generation to learn.

A third possibility would be to broaden the university community so it can draw upon a larger fraction of the existing manpower pool. From a labor management point of view, universities are the most homogeneous, restricted, single-purpose organizations imaginable. They will accept only one type of employee, namely a Ph.D., and have only one kind of job, i.e., a combination of teaching and research. That is traditional, but given the present dilemma one must ask is it necessary? Are there other ways of doing things which would utilize a broader range of talents, provide more diverse career opportunities and hence make universities attractive places of employment for a larger class of people? If so, a stable equilibrium between manpower supply and demand might be reached sooner.

In this connection, it is useful to consider the case of business and management curricula. Fig. 4G shows the number of degrees awarded in business during seven recent years. It is clear that bachelor's and master's degrees are increasing but note, especially, the very small number of Ph.D.s, approximately one doctorate for every 200 bachelor's degrees. This ratio can be sustained because professional curricula have been defined for business which draw upon the course offerings of many parts of the university and do not depend upon the Ph.D. faculty of one discipline to carry the teaching load. Defining a professional curriculum for computer science might be useful, too. Such a curriculum could recognize that many of the undergraduate technical courses in computer science do not need to be taught by Ph.D.s but could be handled equally well by high quality master's level people who do not, necessarily, have a research orientation. If such positions were given first-class status in universities, new career paths would be open and a larger labor pool would be available. Many of the undergraduate and master's degree students might find a professional curriculum as much to their taste as the present core computer science curriculum with a corresponding reduction in number of majors in the latter. This would help bring supply and demand into better balance.

The second most important factor influencing the university environment was found to be research opportunities. In rapidly changing high technology fields it is extremely difficult for an institution's resources to encompass its requirements for capital equipment and operating costs for either education or research. This is a factor the Federal government can influence, however, and NSF has designed its Coordinated Experimental Research Program in computer science specifically for this purpose. As you know, our CER program reaches computer research no matter whether it is located in Schools of Engineering or Schools of Letters and Science and the ten awards made so far are split about half and half. Federal tax legislation can help contribute to the improvement of university environments, also, by encouraging the donation of equipment and money to educational institutions. Legislation passed in the last 18 months is intended to make tax incentives more effective for this purpose.

The third most important factor is salary and this, again, is an institution decision. Differential salary scales which provide preferred treatment for computer science and engineering faculty are becoming commonplace. As a result, it is apparent that the universities are adjusting in this dimension, albeit at the price of some internal stress and discontent.

4.2 *Pre-College Education*

The manpower dilemma of pre-college education is much the same as that of higher education--a dwindling pool of qualified teachers is required to deal with rapidly changing demands for types of training using inflexible, possibly decreasing resources. Also, the working environment is not encouraging qualified teachers to stay in teaching or students to prepare for careers in teaching. To a greater degree than higher education, however, pre-college education may be able to take effective steps to reduce its manpower requirements. The subject matter to be covered has more permanence than high-level college course content does and, as a result, warrants greater investment in curriculum development. Also, more instructional time is devoted to basic knowledge and skills development, areas which lend themselves best to the use of technology in education. Other factors influencing the working environment in pre-college education, "institutional disincentives", professional opportunity, and salary, although different in detail, are very similar to those in higher education, however, but seem even less accessible to constructive intervention by the federal government. Tax incentives to encourage donations of equipment for pre-college education would help accelerate the use of technology and improve the professional opportunities for creative persons but it is not clear that they would influence the supply of qualified teachers very much. Individual institutions or local school districts have jurisdiction over the rest of the relevant factors and it is difficult to imagine what changes they could make which might reverse the current decline in production of qualified teachers or the presently poor competitive position of schools for available talent. Placing schools on an eleven or twelve month calendar might resolve a number of problems but that would be a change so revolutionary to the operation of our whole society that it hardly bears contemplating. As a result, we are almost forced to place our hope in better use of computer technology, itself, to substitute for an irremediable deficiency in educational manpower.

5.0 *CONCLUSION*

The preceding analysis shows that the United States has an imbalance in the supply and demand for computer manpower which is expected to persist for the foreseeable future but that our society is adjusting rapidly to redress that imbalance in every sector except education. In education, the channels for adjusting supply and demand which are available to business, industry and government, principally mobility of personnel among jobs and shifts in the interests of students training for jobs, are ineffective and we cannot hope for short term solutions. This places our educational institutions under great stress and could lead industry to develop alternate methods or new institutions to provide the professional computer personnel it needs if the traditional educational institutions are unable to adjust. One can imagine fundamental changes in the structure and management of institutions, both of higher and pre-college education, which could help resolve the present dilemma but not without altering time-honored traditions. No other technological advance in history has been so rapid or so compelling in confronting education with the hard choice of either embracing fundamental change or accepting a reduction in its traditional role of training students for the future.

REFERENCES

1 John W. Hamblen and Carolyn P. Landis, The Fourth Inventory of Computers in Higher Education: An Interpretive Report, *EDUCOM Series in Computing and Telecommunications in Higher Education*, 4, (1980).

2 See, e.g., Taulbee, O.E., and Conte, S.D. Production and Employment of Ph.D.'s in Computer Science--1977 and 1978. *Comm. ACM 22*, 2 (Feb. 1979), 75-76.

3 In this connection, see also Fairley, Richard E. Employment Characteristics of Doctoral Level Computer Scientists. *Comm. ACM 22*, 2 (Feb. 1979), 77-78.

4 A workshop organized in April, 1982 by Prof. E. Dubinsky, Dept. of Mathematics, Clarkson College, considered the question of retraining mathematicians to teach the core undegraduate curriculum in computer science. An experimental plan was developed which was believed to be worthy of a trial. It is noteworthy, that even though the intellectual training and experience a professional mathematician brings to the task of retraining himself in computer science is superior, a minimum of two summers plus an intermediate year of in service training was considered necessary. It was believed this would create an adequate teacher of undergraduate courses but not an academic computer scientist.

Ratio of Ph.D's

to

Bachelor's Degrees

(1980)

Physics	Chemistry	Mathematics	Engineering	Computer Science
1:4	1:7	1:15	1:18	1:46

Figure 1

Ratio of Ph.D's

to

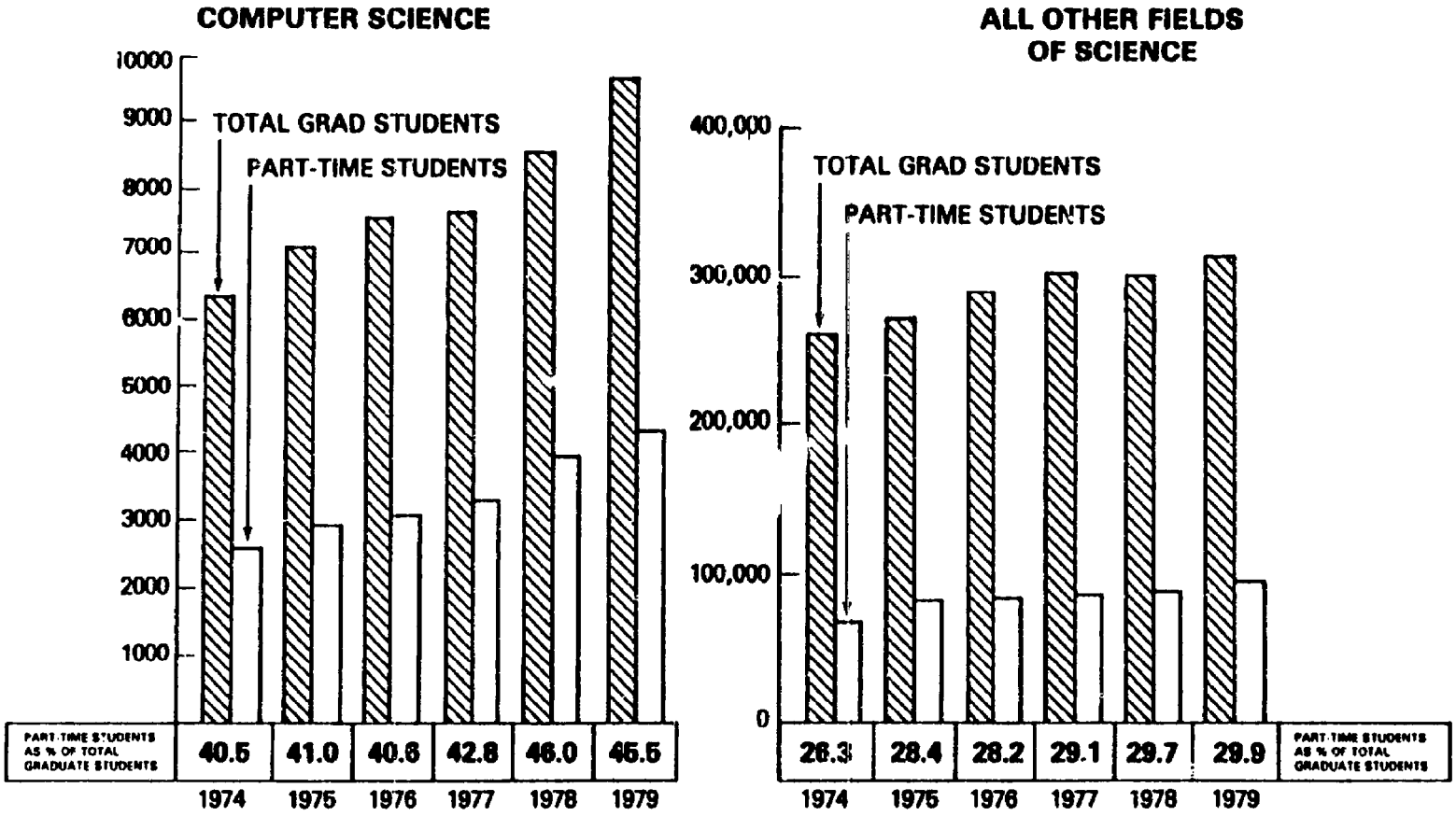
Master's Degrees

(1980)

Physics	Chemistry	Mathematics	Engineering	Computer Science
5:7	10:11	1:4	1:6	1:15

Figure 2

PART-TIME GRADUATE STUDENTS IN COMPUTER SCIENCE



SOURCE: ACADEMIC SCIENCE; GRADUATE ENROLLMENT AND SUPPORT (FALL 1979)

Figure 3

MCS 81-2842
8-4-81

FACULTY EMPLOYMENT CHANGES

	NO.	%
UNIVERSITY TO UNIVERSITY	46	66
UNIVERSITY TO INDUSTRY	24	34

Figure 4

Full Time Engineering Faculty Moving To

Industry After The 1979/1980 Academic Year

Computer	5.6%
Industrial	2.8
Electrical	2.8
Mechanical	2.7
Chemical	2.6
Civil	2.3
Aeronautical	1.9

Figure 5

NEW C.S. Ph.D EMPLOYMENT

	NO.	%
UNIVERSITY	70	55
INDUSTRY	57	44
NON C.S. POSITION	1	1
TOTAL	128	100

Figure 6

REASONS FOR LEAVING

THE UNIVERSITIES FOR INDUSTRY

A. INSTITUTIONAL DISINCENTIVES

1. HEAVY TEACHING LOADS

2. JOB INSECURITY

3. HASSLE GETTING SUPPORT AND PERQUISITES

B. POOR RESEARCH FACILITIES

C. SALARY

Figure 7

INCREASE IN STARTING

SALARIES (1974-1979)

COMPUTER PROFESSIONALS	MATHEMATICS AND CHEMISTRY	AGRICULTURE, SOCIAL SCIENCES, BUSINESS, BIOLOGY AND HUMANITIES
53-58%	51%	33-42%

Figure 8

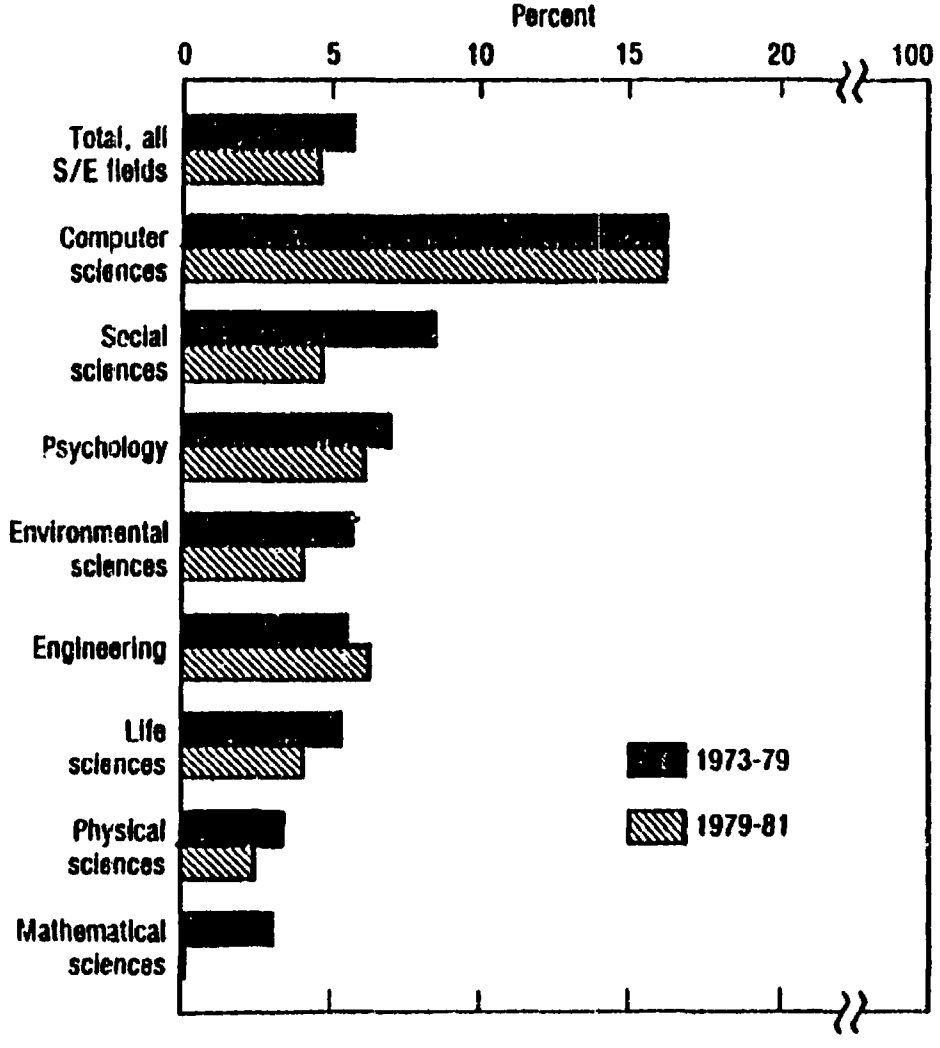
UNEMPLOYMENT RATES

(1978)

COMPUTER PROFESSIONALS	OTHER SCIENCE AND ENGINEERING	TOTAL CIVILIAN
0.3%	1.5%	6.0%

Figure 9

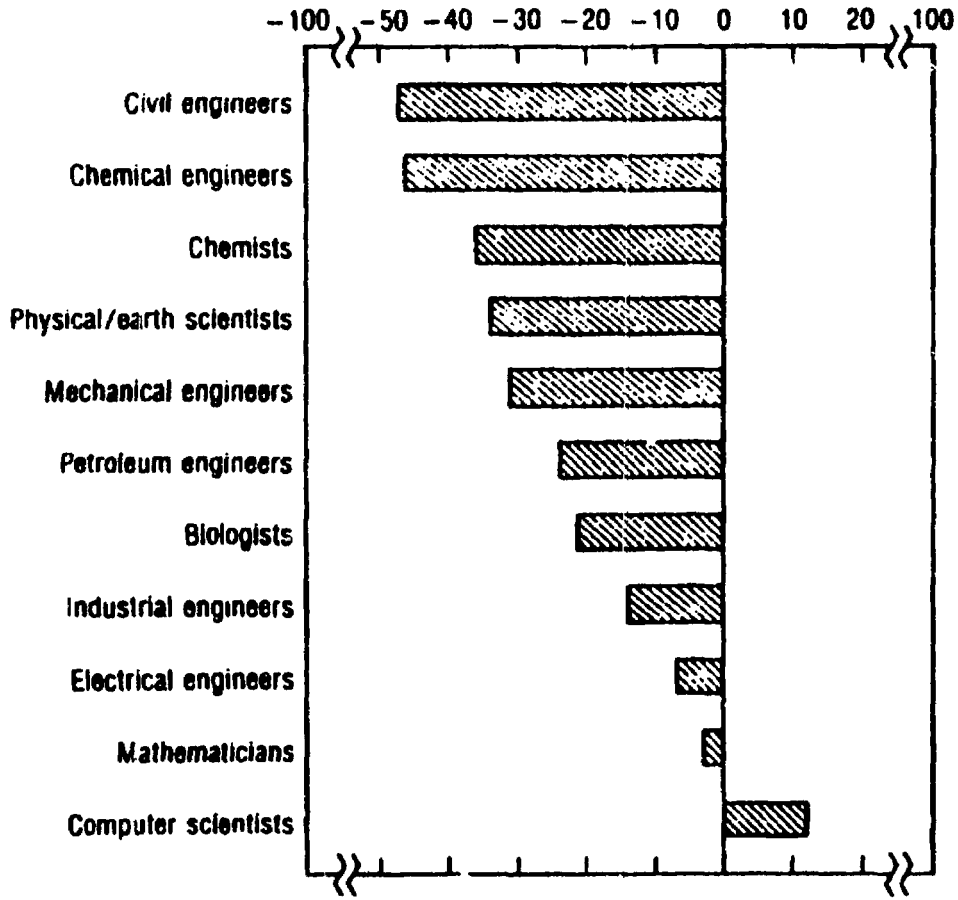
Average annual Rate of growth of science/ engineering (S/E)-employed Ph.D.'s by field: 1973-79 and 1979-81



SOURCE: National Science Foundation

Figure 10

Percent changes in job offers to bachelor's-level science and engineering graduates between July 1981 and July 1982



SOURCE: College Placement Council

Figure 11

PROJECT DEMAND/SUPPLY BALANCE

IN SCIENTIFIC AND ENGINEERING OCCUPATIONS: 1987

SURPLUS OR SHORTAGE OCCUPATIONS HAVE SUPPLY ESTIMATES WHICH DEVIATE FROM PROJECTED DEMAND BY 10% OR MORE; POTENTIAL SURPLUS OR SHORTAGE OCCUPATIONS HAVE AN ESTIMATED SUPPLY OF WORKERS DEVIATING FROM THE PROJECTED LEVEL OF DEMAND BY 5% TO 10%; A BALANCE IS PROJECTED WHEN AVAILABLE SUPPLY IS WITHIN 5% OF DEMAND IN EITHER DIRECTION.

	LOW ECONOMIC GROWTH/LOW DEFENSE	HIGH ECONOMIC GROWTH/HIGH DEFENSE
SCIENTISTS		
AGRICULTURAL	SURPLUS	SURPLUS
BIOLOGISTS	SURPLUS	SURPLUS
CHEMISTS	SURPLUS	SURPLUS
COMPUTER SPECIALISTS[1]	SHORTAGE	SHORTAGE
GEOLOGISTS	SURPLUS	SURPLUS
MATHEMATICIANS	SURPLUS	SURPLUS
PHYSICISTS	SURPLUS	SURPLUS
SOCIAL	SURPLUS	SURPLUS
ENGINEERS		
AERONAUTICAL	SHORTAGE	SHORTAGE
CHEMICAL	SURPLUS	SURPLUS
CIVIL	SURPLUS	SURPLUS
ELECTRICAL/ELECTRONIC	BALANCE	POTENTIAL SHORTAGE
INDUSTRIAL	POTENTIAL SURPLUS	BALANCE
MECHANICAL	SURPLUS	POTENTIAL SURPLUS
METALLURGICAL	SURPLUS	SURPLUS
MINING/PETROLEUM	SURPLUS	SURPLUS
ENGINEERING, N.E.C.	SURPLUS	SURPLUS

[1]Includes both computer systems analysts and programmers.

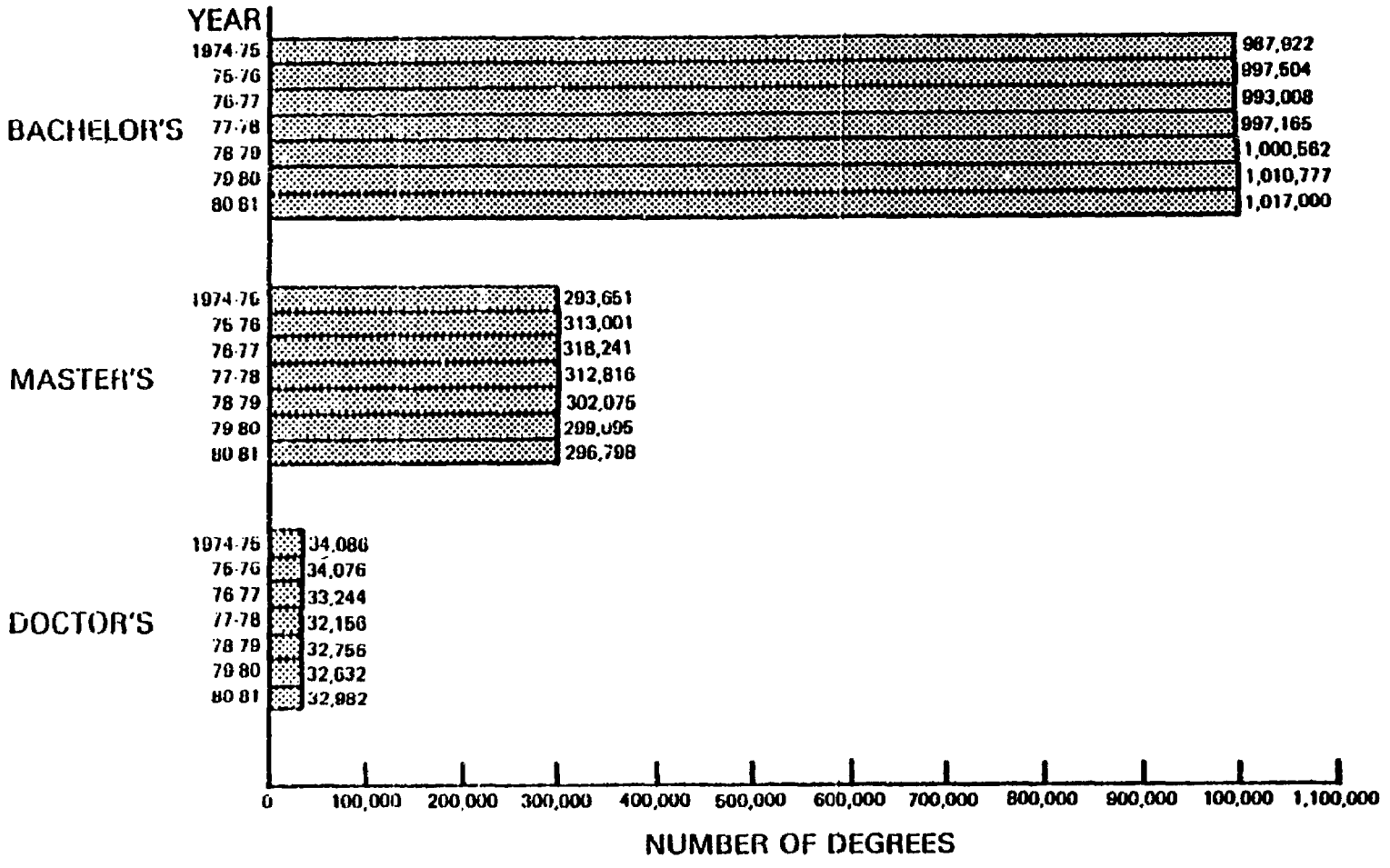
Source: National Science Foundation

Figure 12

	Share of World Trade		Engineering Graduates As A Proportion of Relevant Age Group
	1963	1977	(1977-78)
United Kingdom	15%	9%	1.7%
West Germany	20%	21%	2.3%
Japan	8%	15%	4.2%
United States	21%	16%	1.6%

Figure 13

TOTAL DEGREES AWARDED
(ALL FIELDS)



SOURCES: NATIONAL SCIENCE FOUNDATION
NATIONAL CENTER FOR EDUCATION STATISTICS

MCS 83-447
1-5-83

Figure 14

ANNUAL PROJECTIONS

OF

18 YEAR OLDS IN U.S.

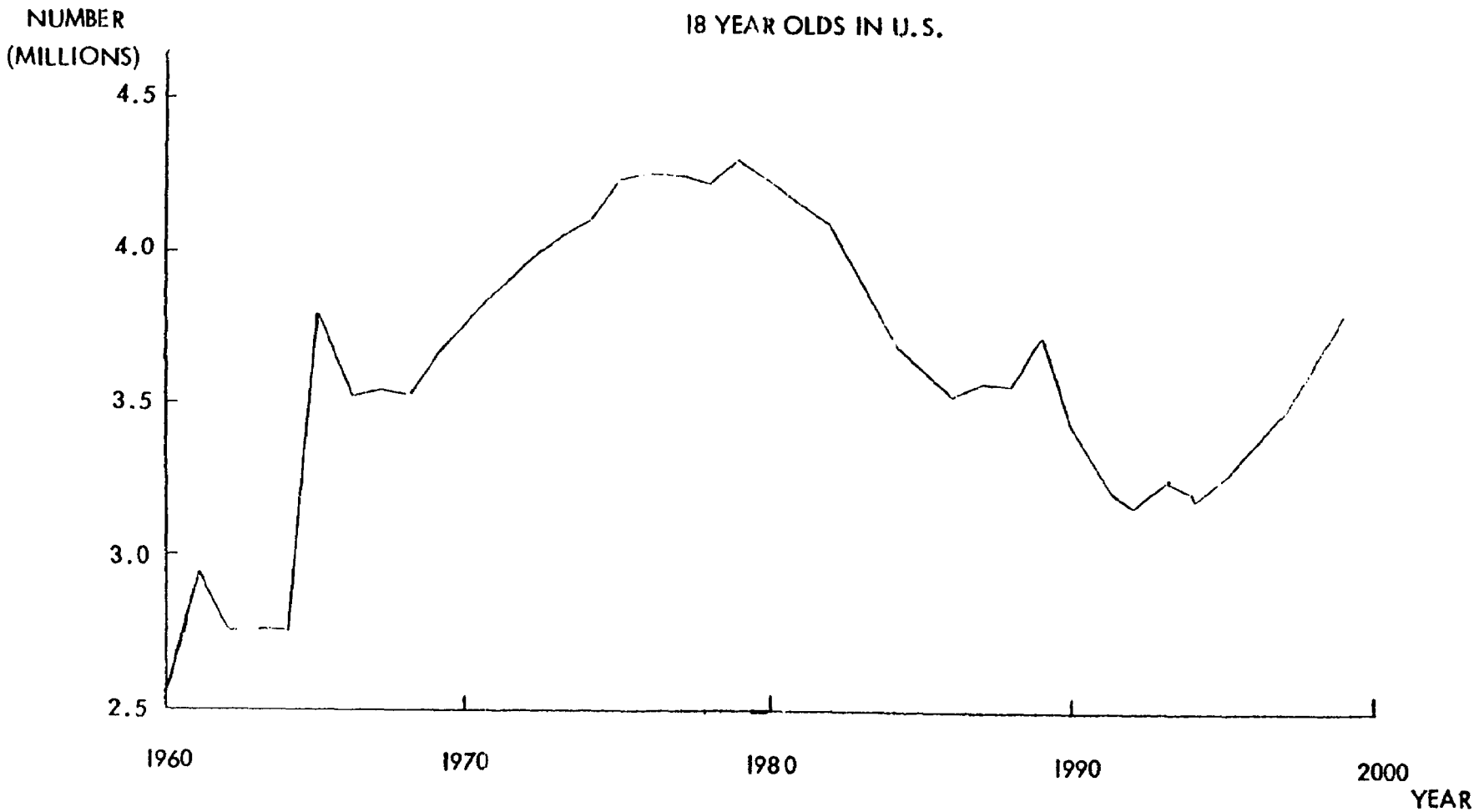
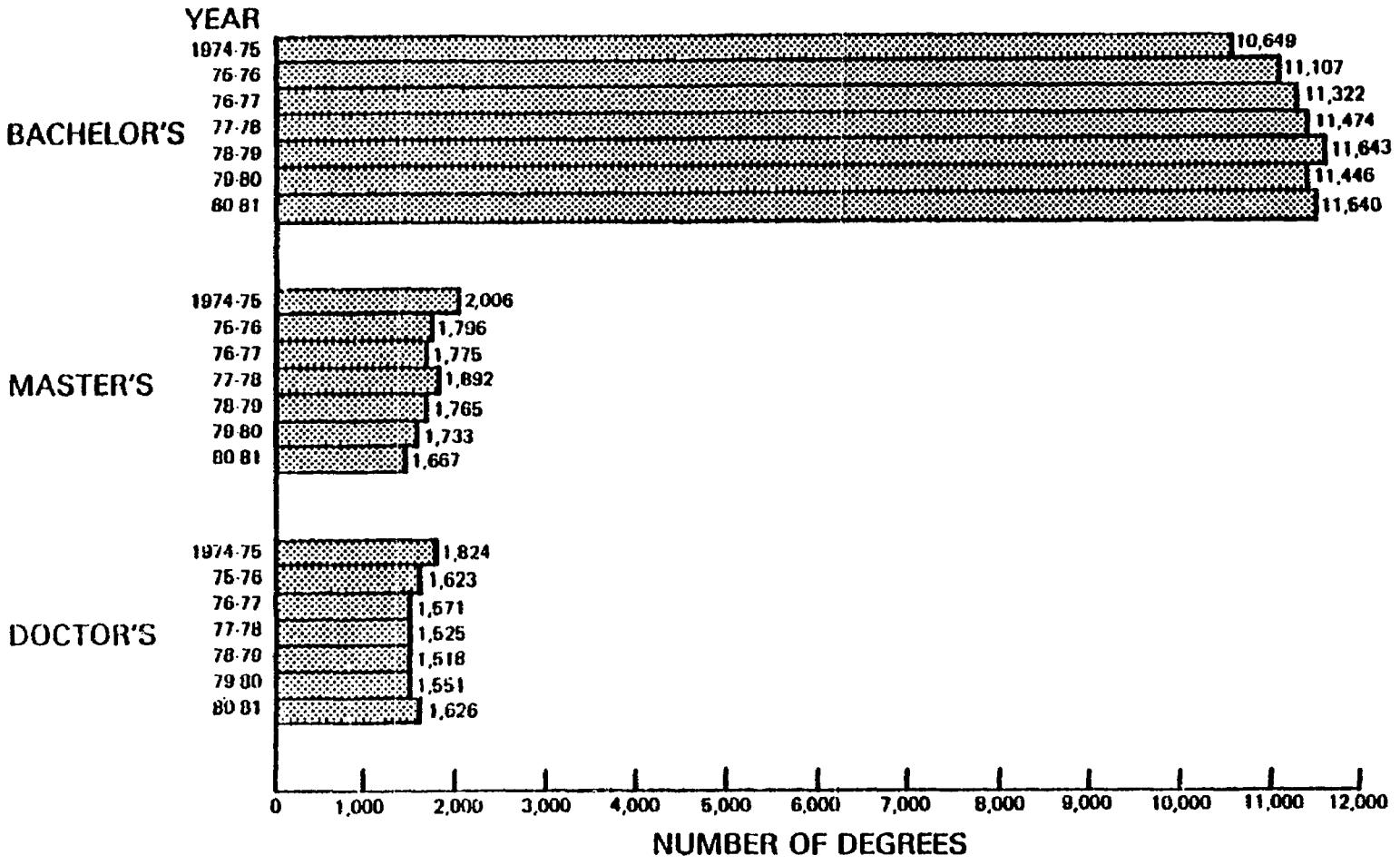


Figure 15

DEGREES AWARDED IN CHEMISTRY

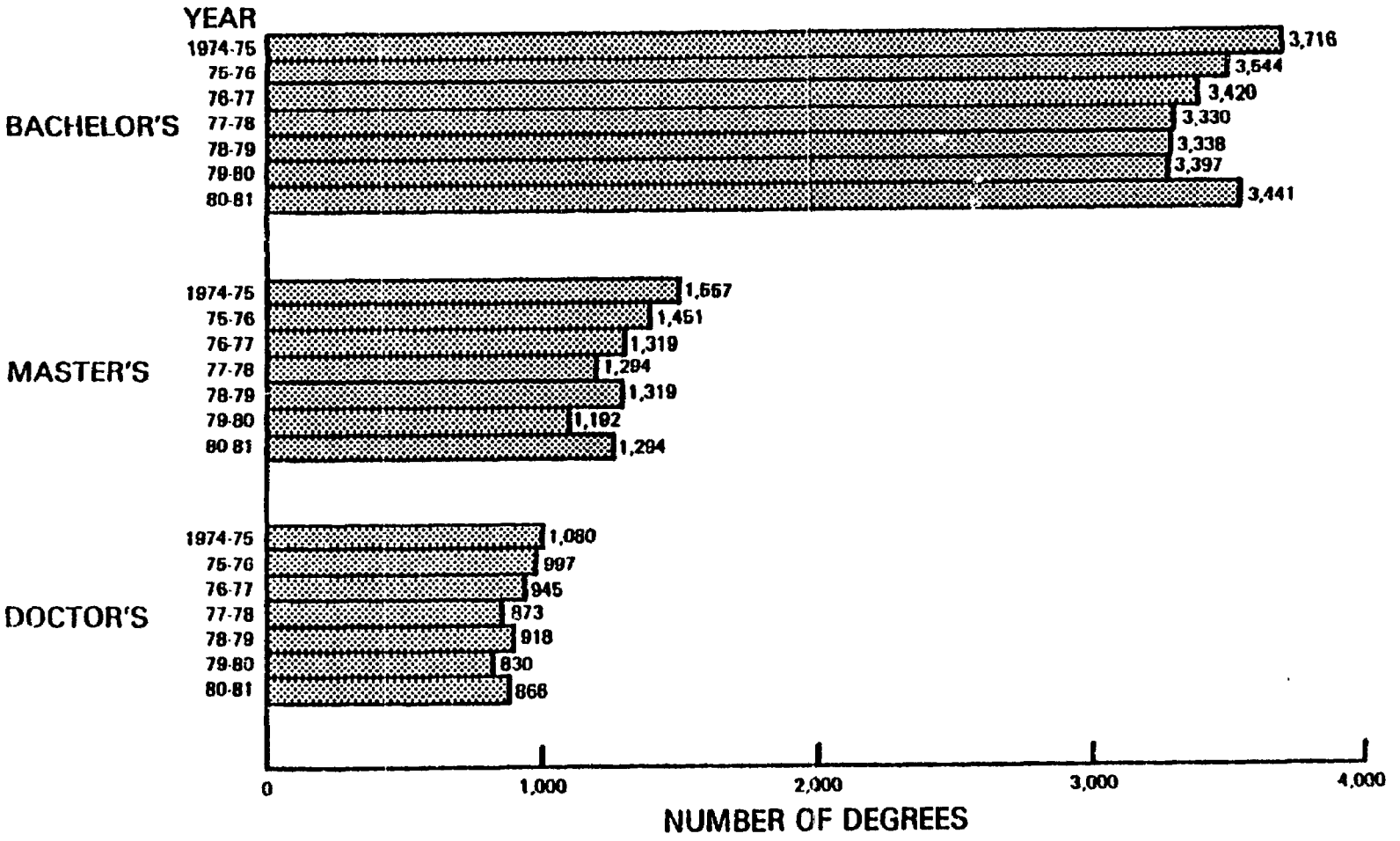


SOURCES: NATIONAL SCIENCE FOUNDATION
NATIONAL CENTER FOR EDUCATION STATISTICS

Figure 16

MCS 83-463
1-5-83

DEGREES AWARDED IN PHYSICS

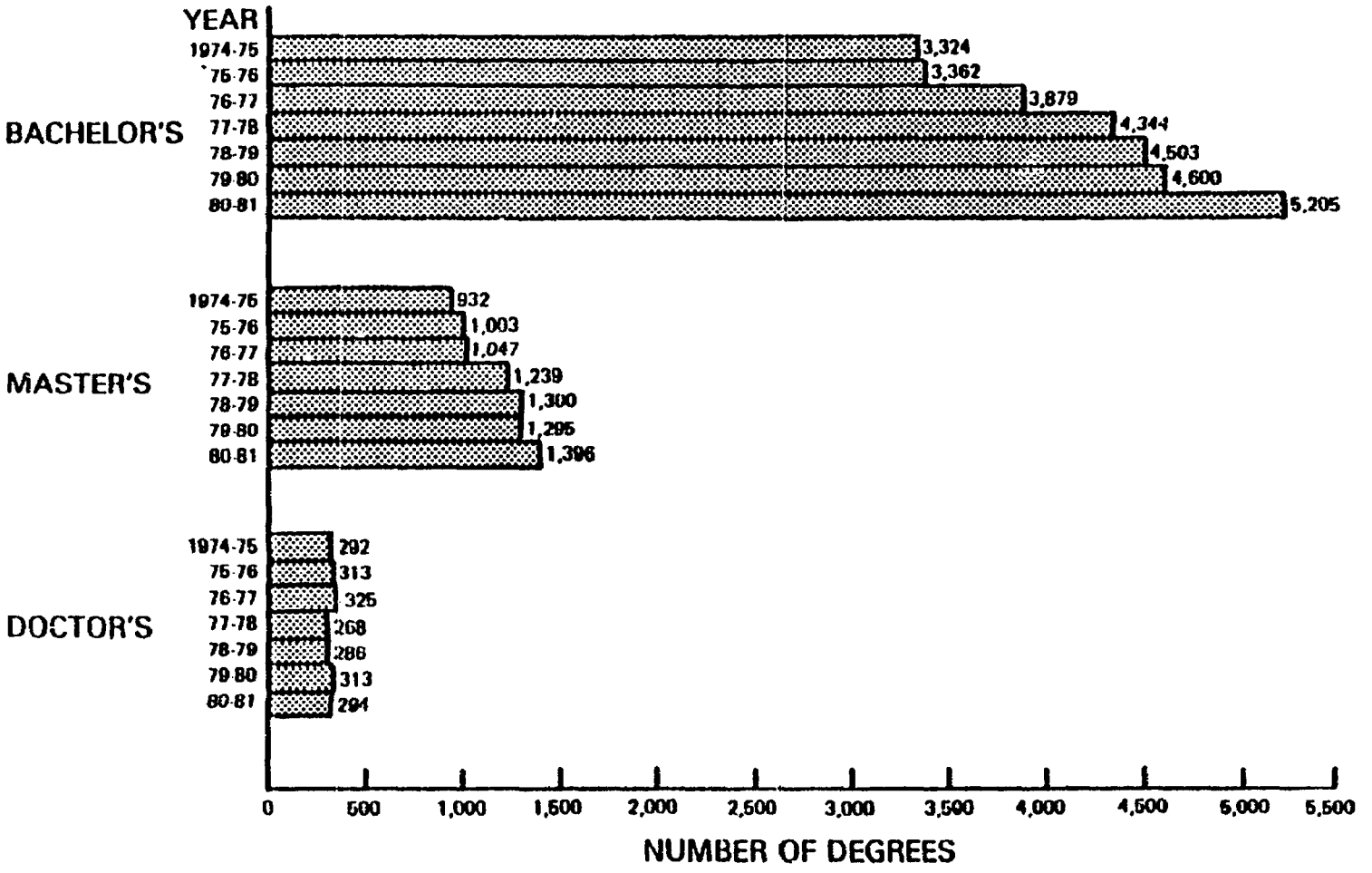


SOURCES: NATIONAL SCIENCE FOUNDATION
NATIONAL CENTER FOR EDUCATION STATISTICS

Figure 17

MCS 83-462
1-6-83

DEGREES AWARDED IN GEOLOGICAL SCIENCES

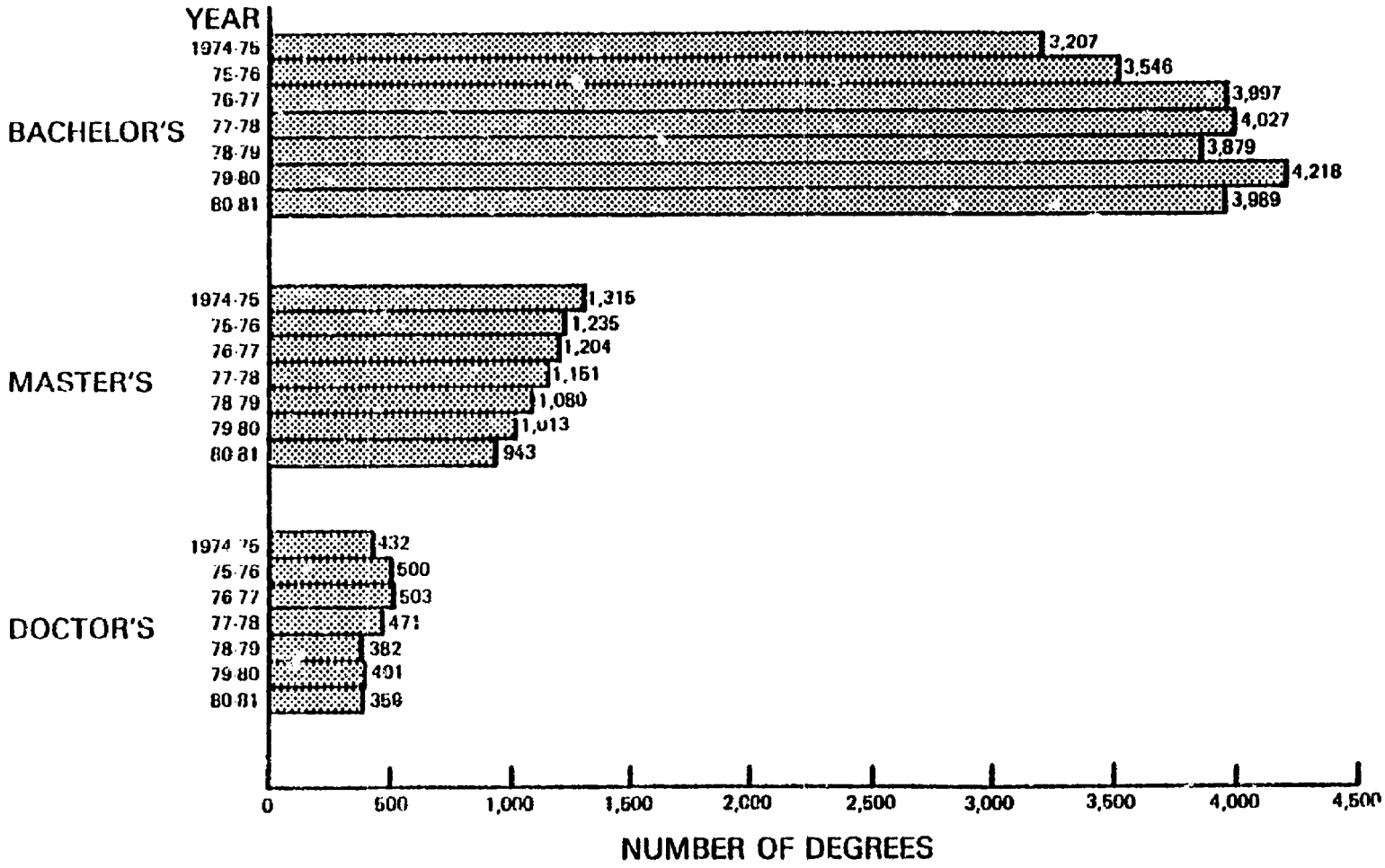


SOURCES: NATIONAL SCIENCE FOUNDATION
NATIONAL CENTER FOR EDUCATION STATISTICS

MCS 83-455
1-83

Figure 18

DEGREES AWARDED IN OTHER PHYSICAL SCIENCES

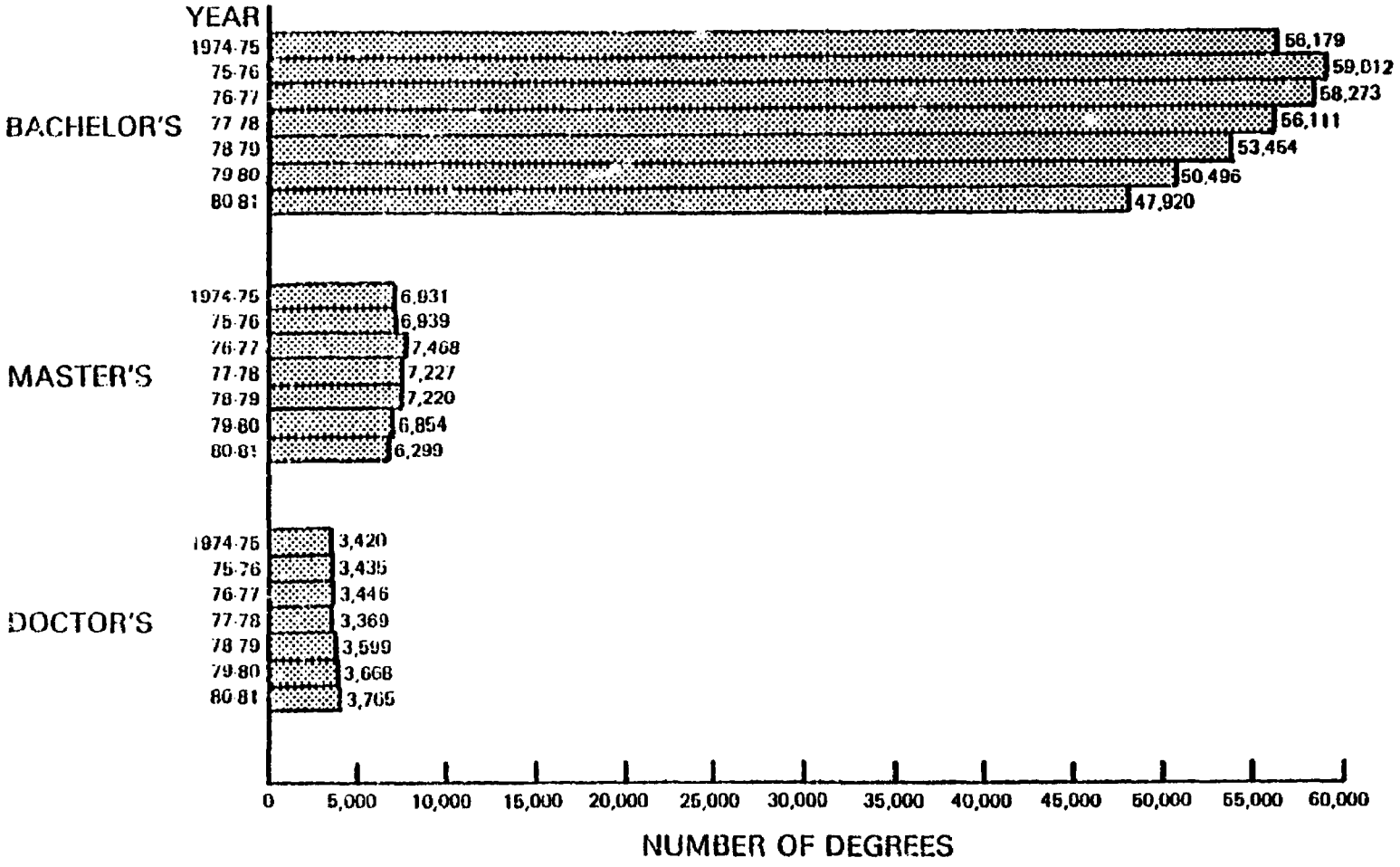


SOURCES: NATIONAL SCIENCE FOUNDATION
NATIONAL CENTER FOR EDUCATION STATISTICS

MCS 83-446
1-6-83

Figure 19

DEGREES AWARDED IN BIOLOGICAL SCIENCES

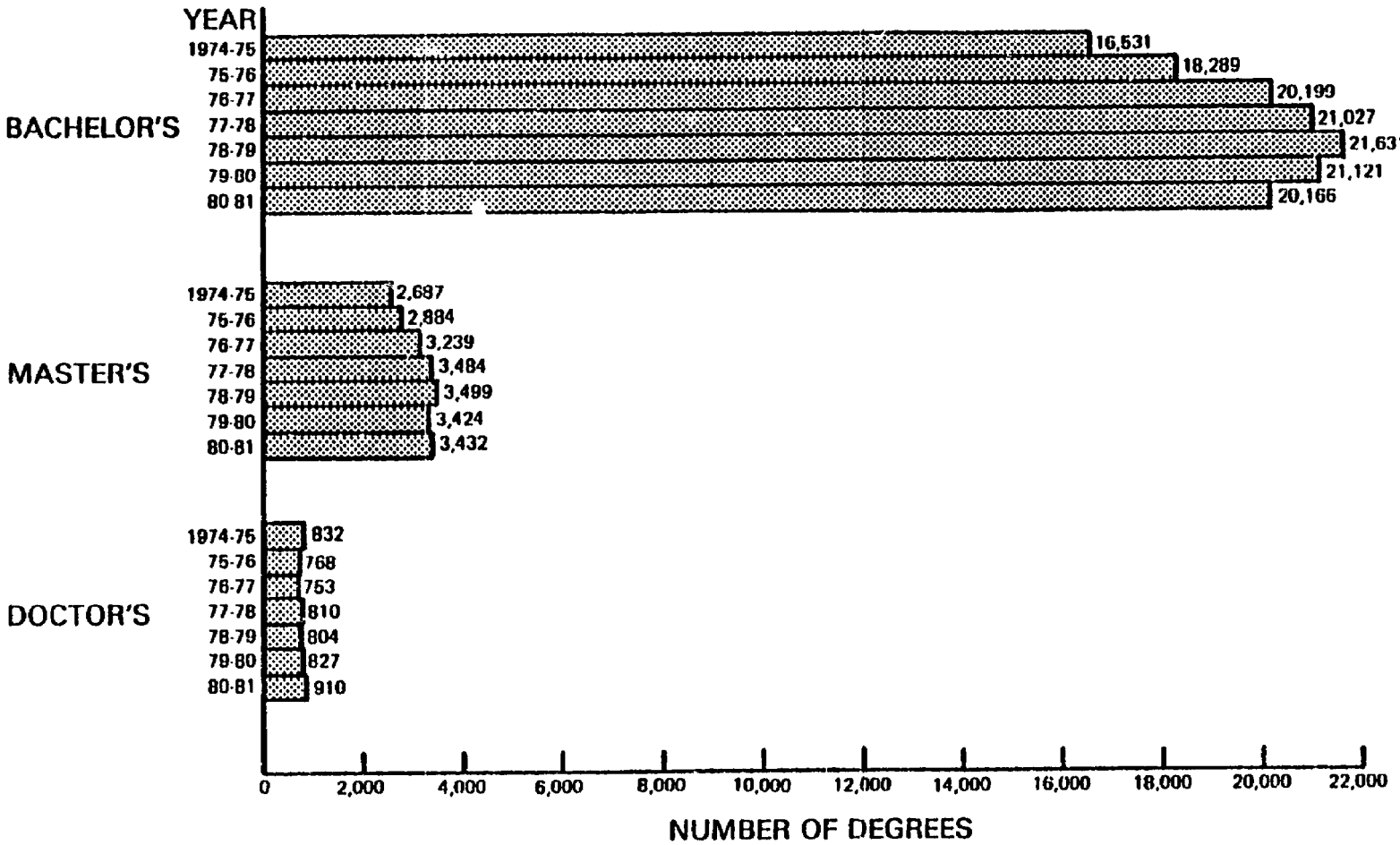


SOURCES: NATIONAL SCIENCE FOUNDATION
NATIONAL CENTER FOR EDUCATION STATISTICS

MCS 83-444
1-5-83

Figure 20

DEGREES AWARDED IN AGRICULTURAL AND NATURAL RESOURCES

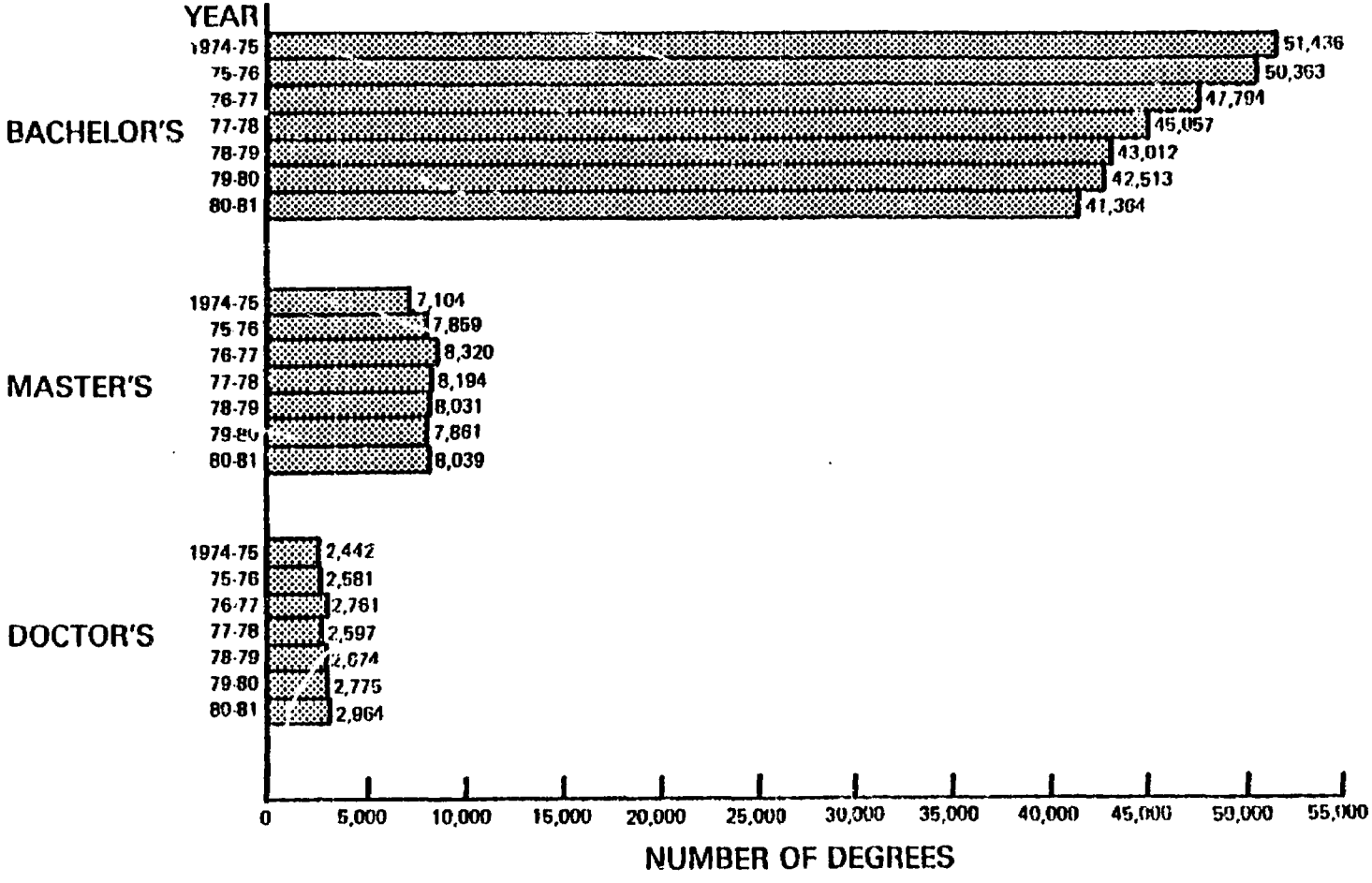


SOURCES: NATIONAL SCIENCE FOUNDATION
NATIONAL CENTER FOR EDUCATION STATISTICS

MCS 83-445
1-5-83

Figure 21

DEGREES AWARDED IN PSYCHOLOGY

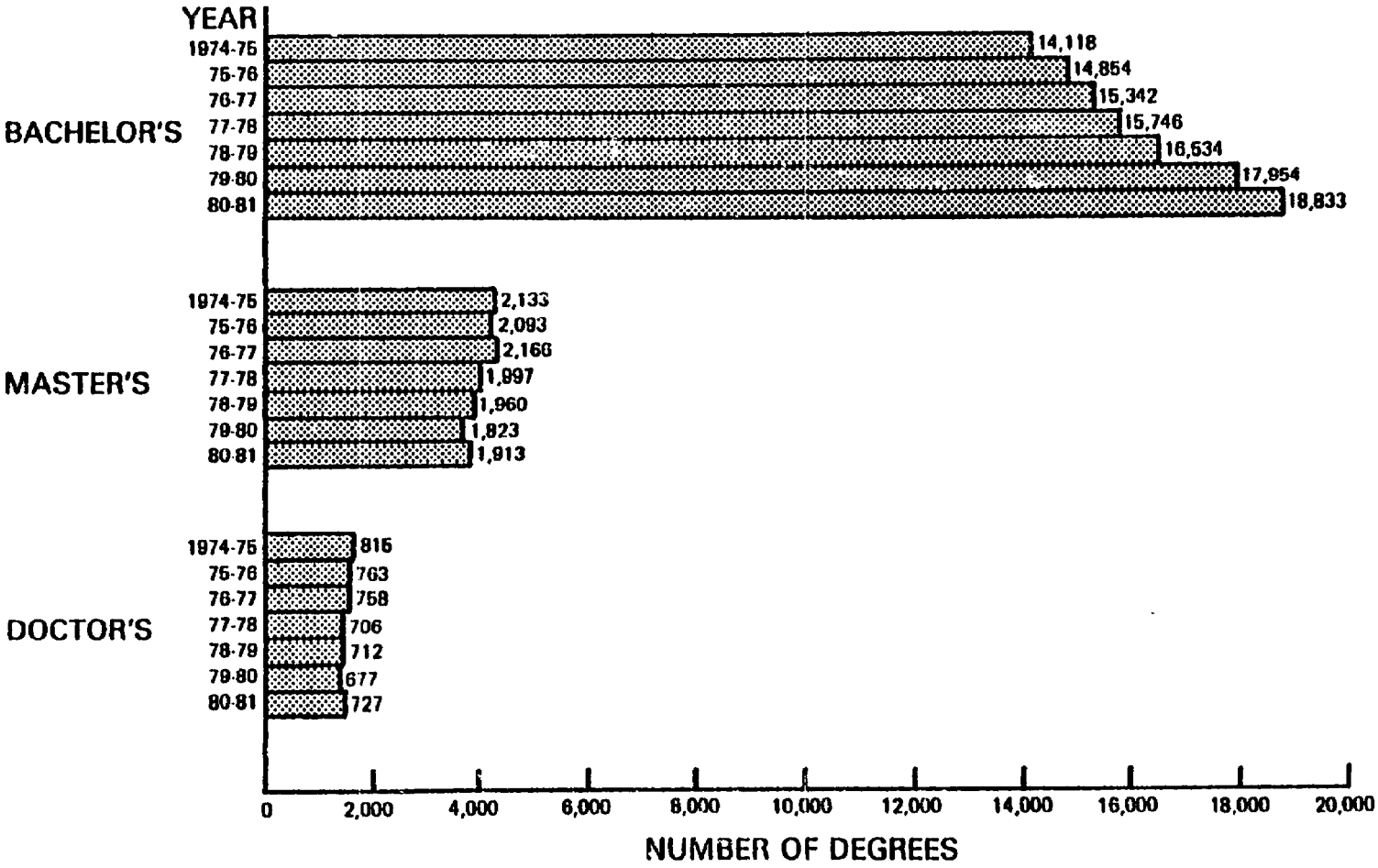


SOURCES: NATIONAL SCIENCE FOUNDATION
NATIONAL CENTER FOR EDUCATION STATISTICS

MCS 83-442
1-5-83

Figure 22

DEGREES AWARDED IN ECONOMICS

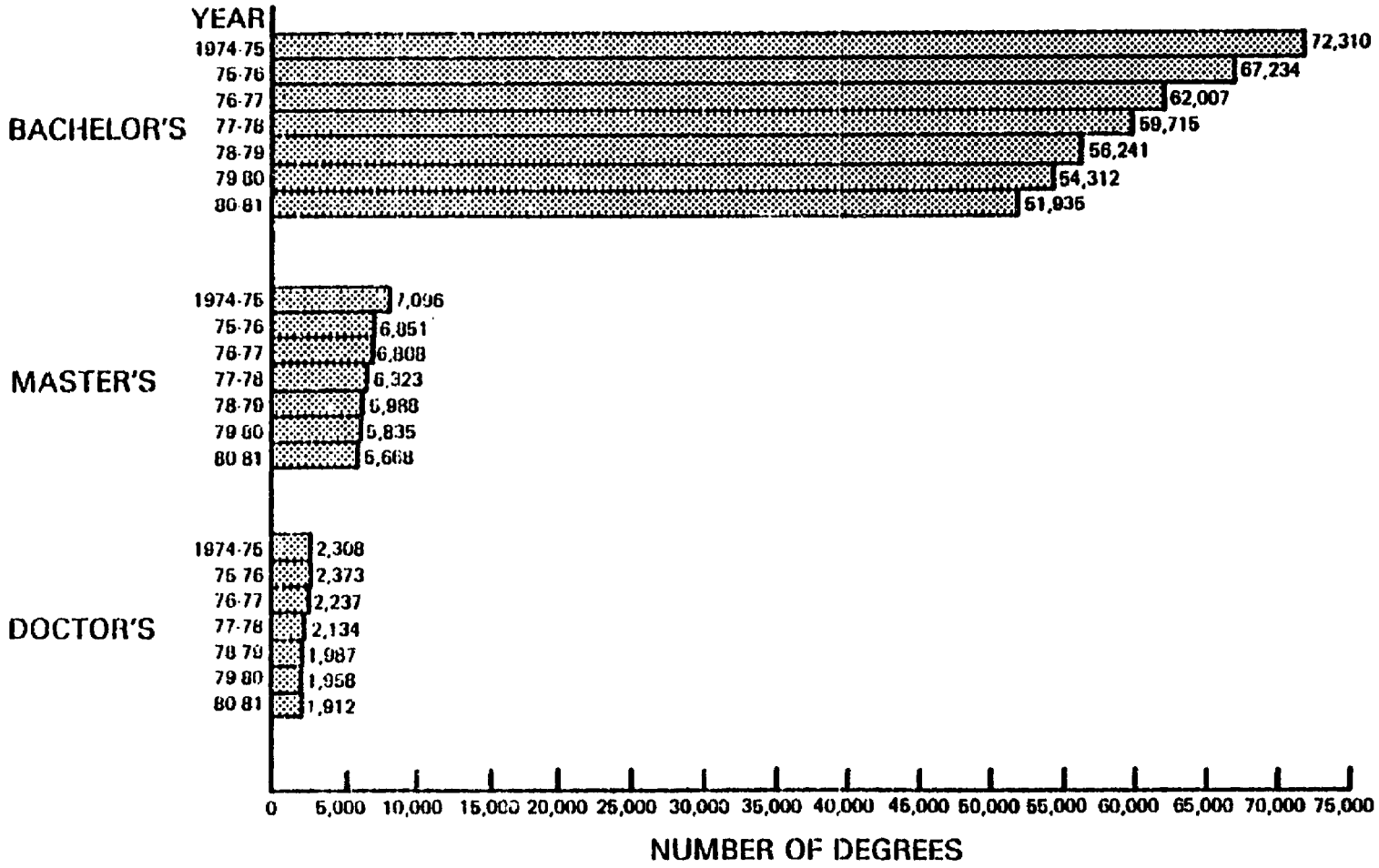


SOURCES: NATIONAL SCIENCE FOUNDATION
NATIONAL CENTER FOR EDUCATION STATISTICS

MCS 83-443
1 5 83

Figure 23

DEGREES AWARDED IN OTHER SOCIAL SCIENCES

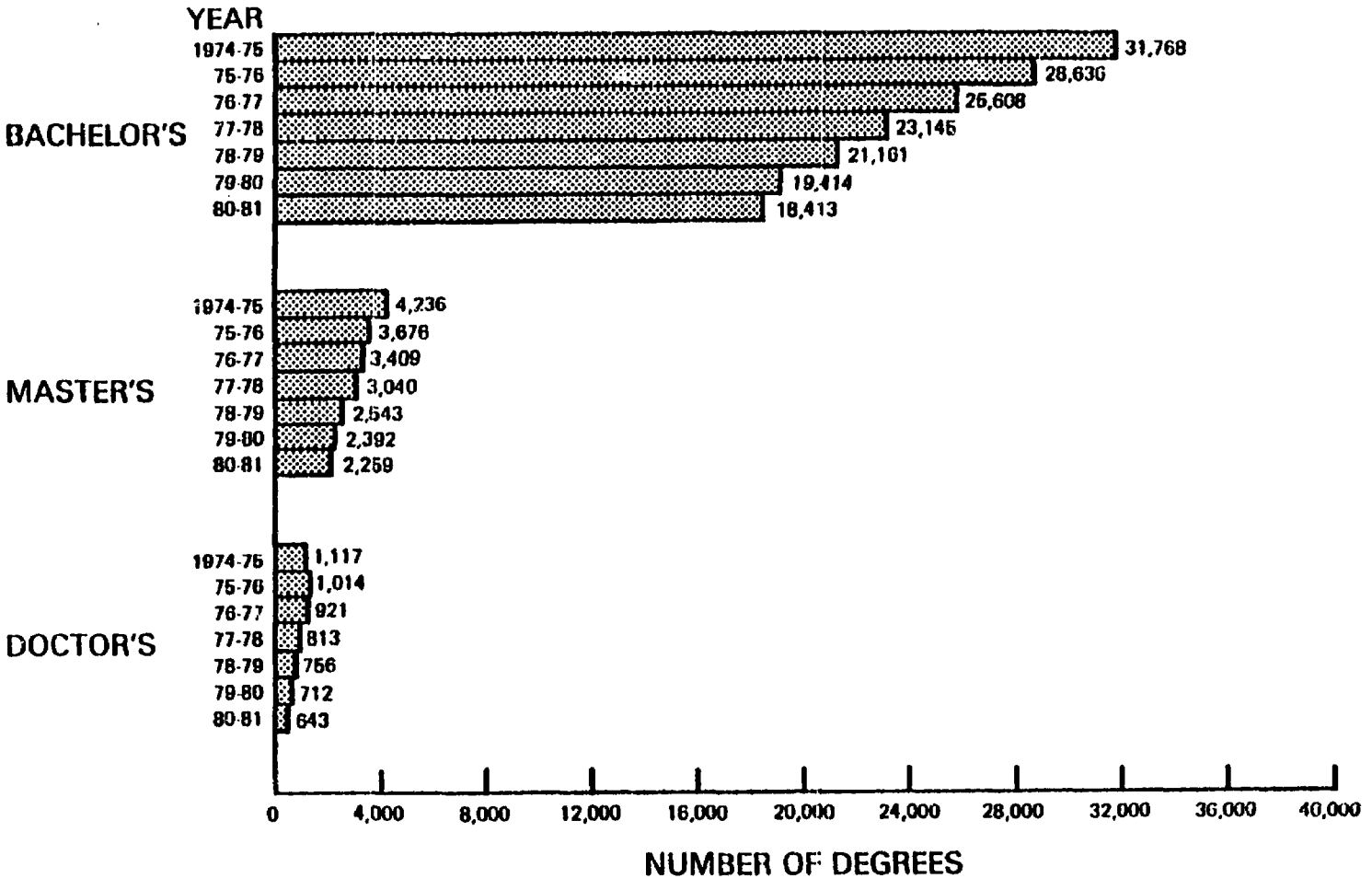


SOURCES: NATIONAL SCIENCE FOUNDAT ON
NATIONAL CENTER FOR EDUCATION STATISTICS

Figure 24

MCS 83-441
1-5-83

DEGREES AWARDED IN HISTORY

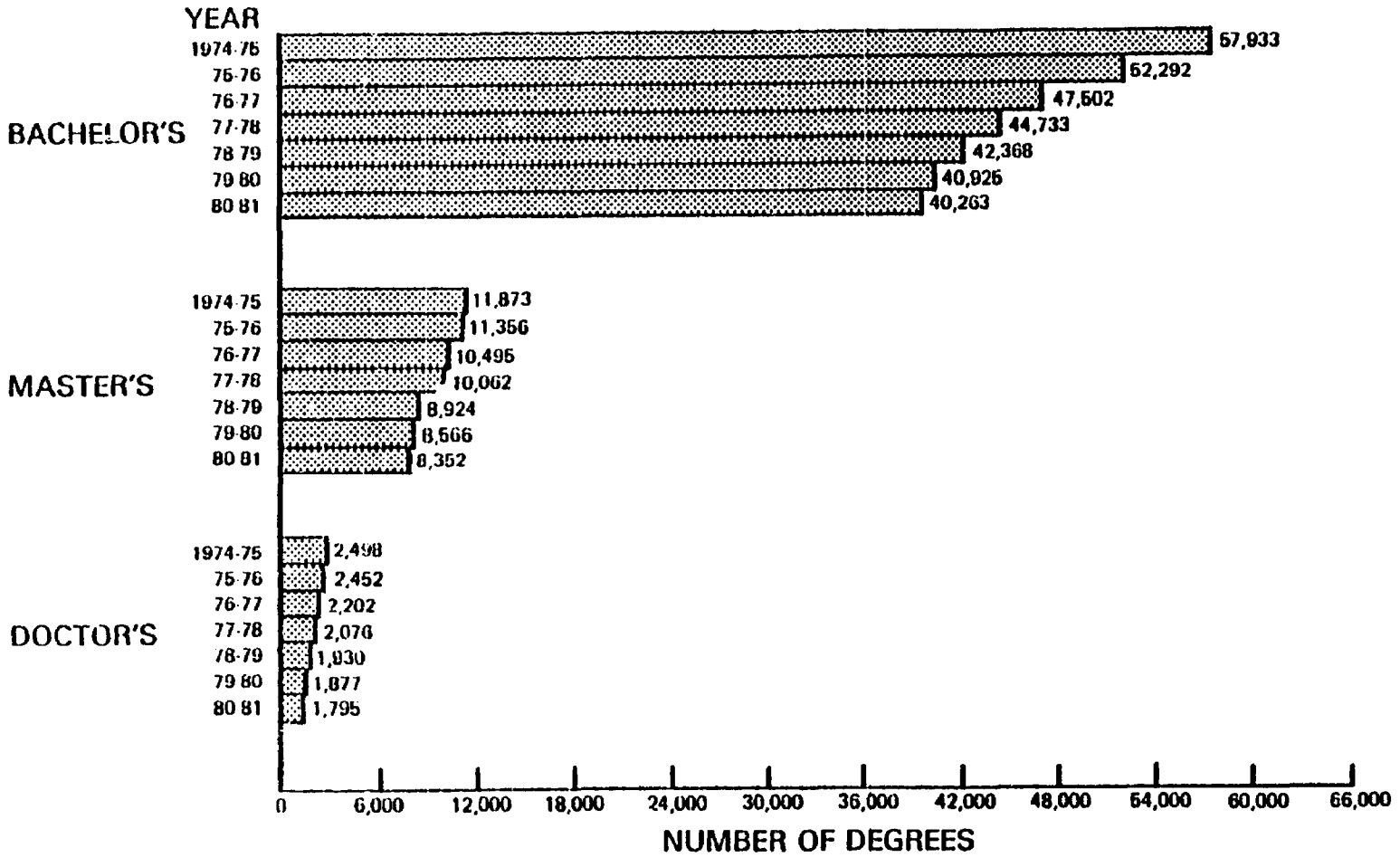


SOURCES: NATIONAL SCIENCE FOUNDATION
NATIONAL CENTER FOR EDUCATION STATISTICS

Figure 25

MCS 83-456
1-5-83

DEGREES AWARDED IN LETTERS[1]



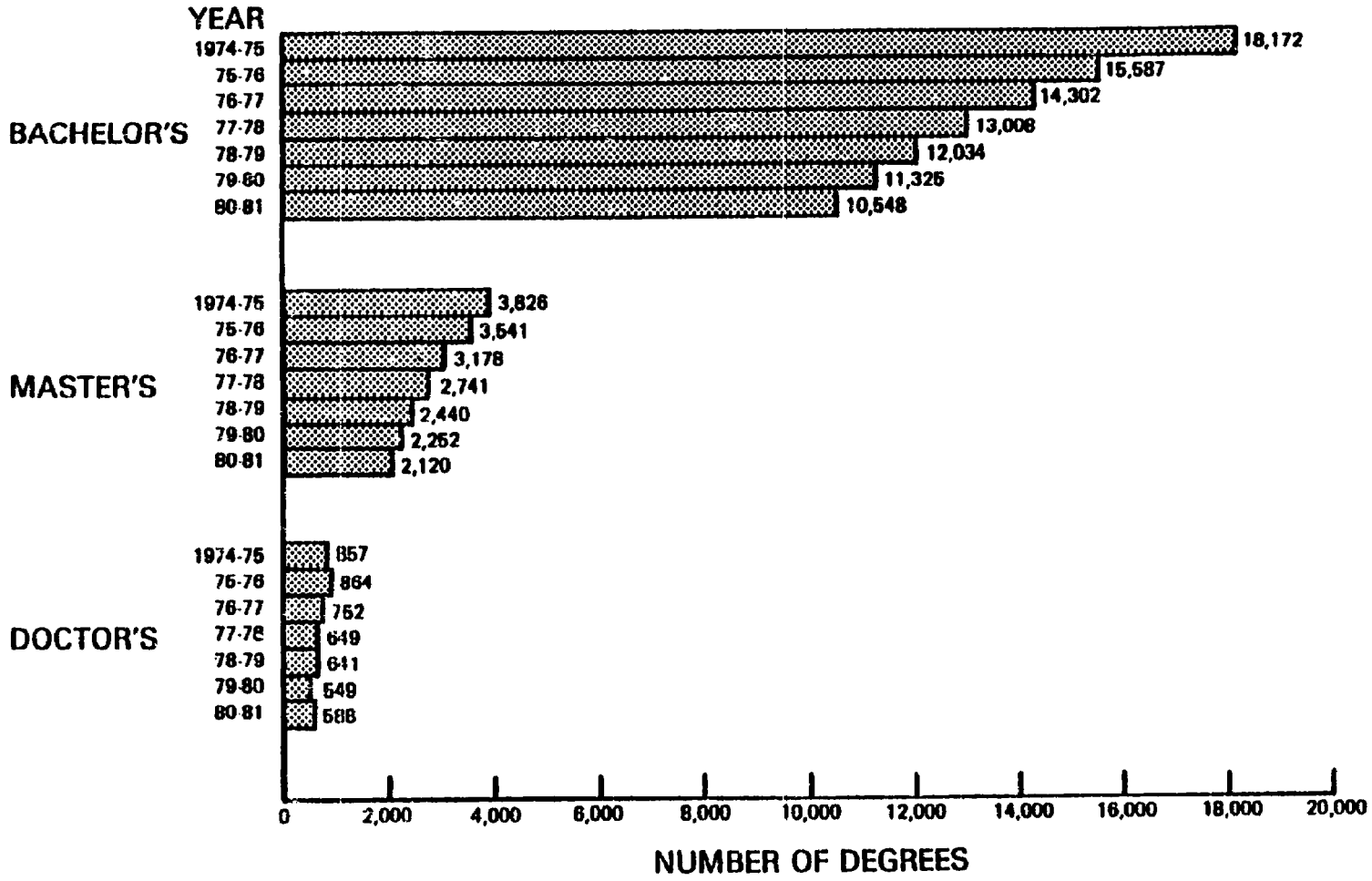
[1]INCLUDES: ENGLISH (GENERAL), LITERATURE (ENGLISH), COMPARATIVE LITERATURE, CLASSICS, LINGUISTICS, SPEECH, DEBATE, AND FORSENIC SCIENCE, CREATIVE WRITING, AND TEACHING OF ENGLISH AS A FOREIGN LANGUAGE.

SOURCES: NATIONAL SCIENCE FOUNDATION
NATIONAL CENTER FOR EDUCATION STATISTICS

Figure 26

MCS 83-457
1-5-83

DEGREES AWARDED IN FOREIGN LANGUAGES

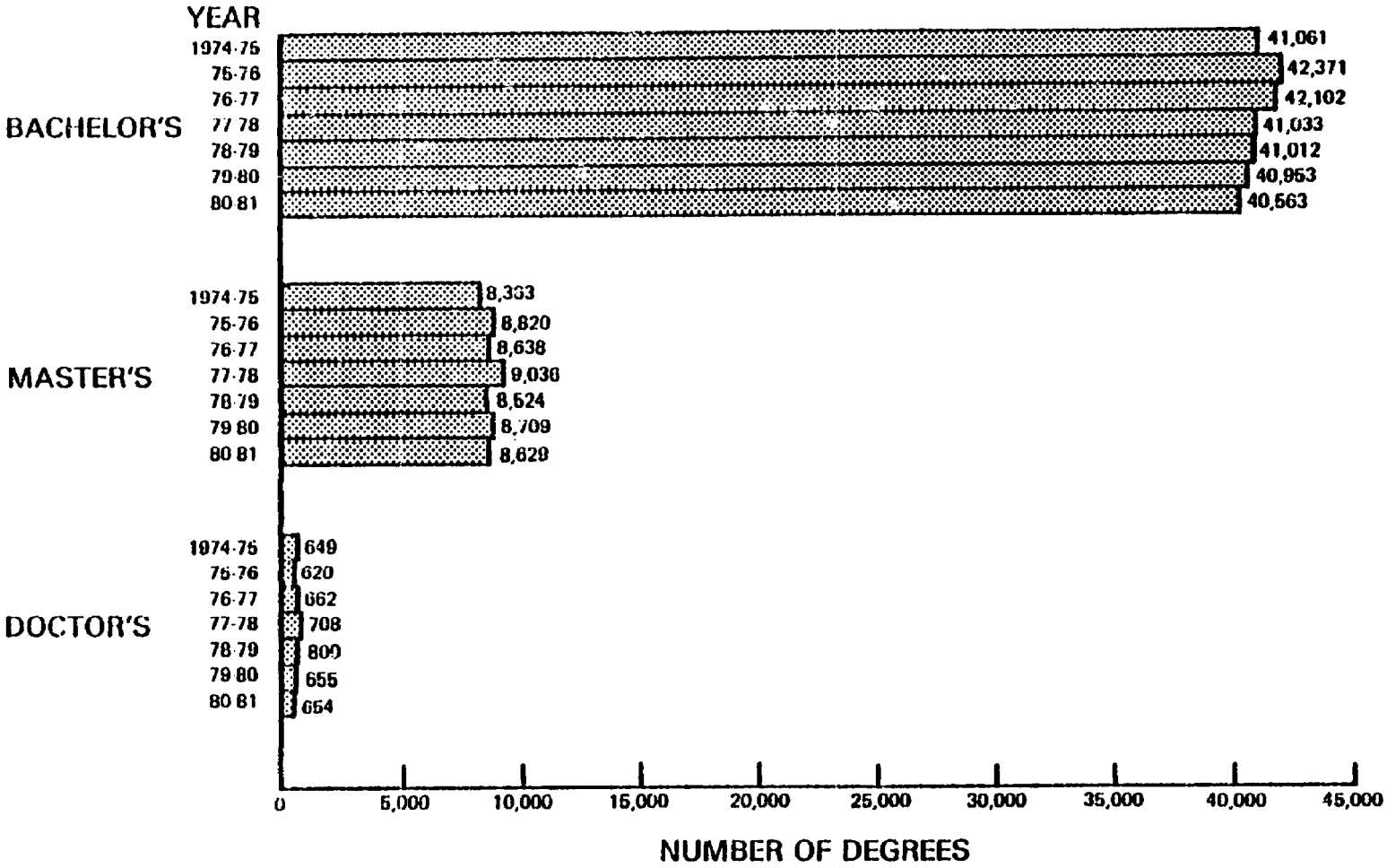


SOURCES: NATIONAL SCIENCE FOUNDATION
NATIONAL CENTER FOR EDUCATION STATISTICS

Figure 27

MCS 83-458
1-5-83

DEGREES AWARDED IN FINE AND APPLIED ARTS[1]



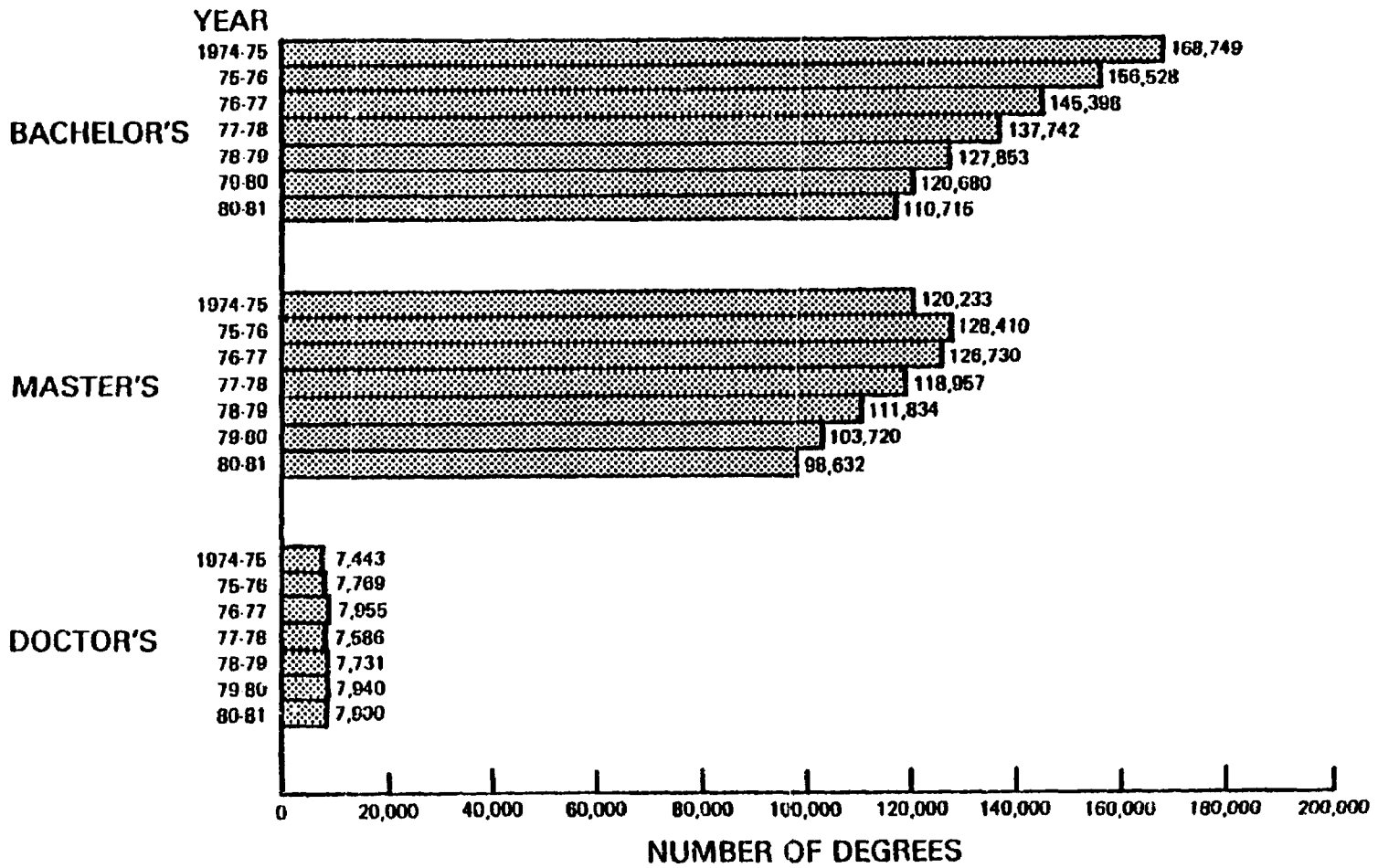
[1]INCLUDES: FINE ARTS (GENERAL), ART, ART HISTORY AND APPRECIATION, MUSIC (PERFORMING, COMPOSITION AND THEORY), MUSIC (LIBERAL ARTS PROGRAM), DRAMATIC ARTS, DANCE, APPLIED DESIGN, CINEMATOGRAPHY, PHOTOGRAPHY, AND OTHER.

SOURCES: NATIONAL SCIENCE FOUNDATION
NATIONAL CENTER FOR EDUCATION STATISTICS

Figure 28

MCS 83-459
1-5-83

DEGREES AWARDED IN EDUCATION

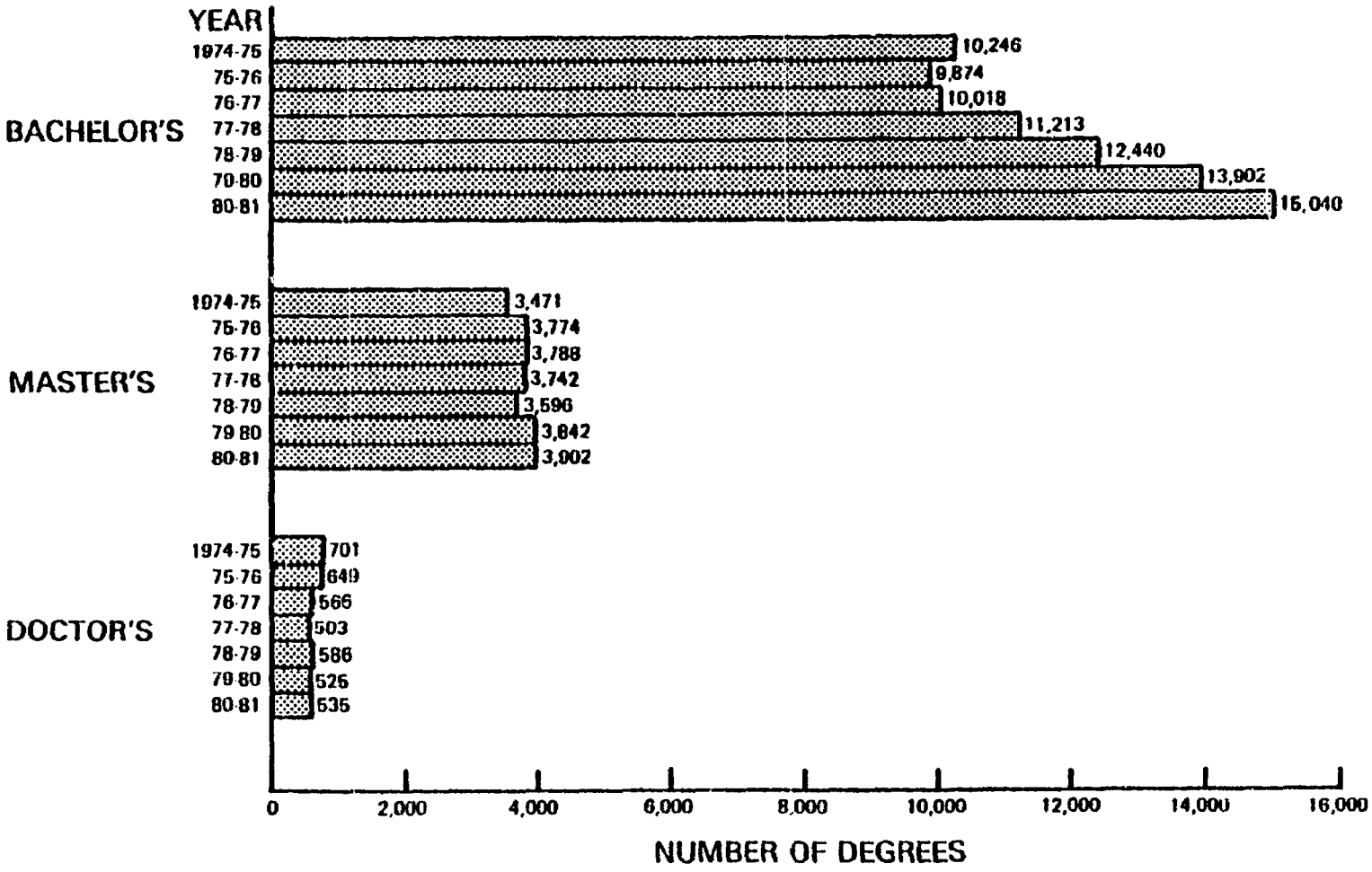


SOURCES: NATIONAL SCIENCE FOUNDATION
NATIONAL CENTER FOR EDUCATION STATISTICS

Figure 29

MCS 83 460
1-6 83

DEGREES AWARDED IN ELECTRICAL, ELECTRONICS AND COMMUNICATIONS ENGINEERING

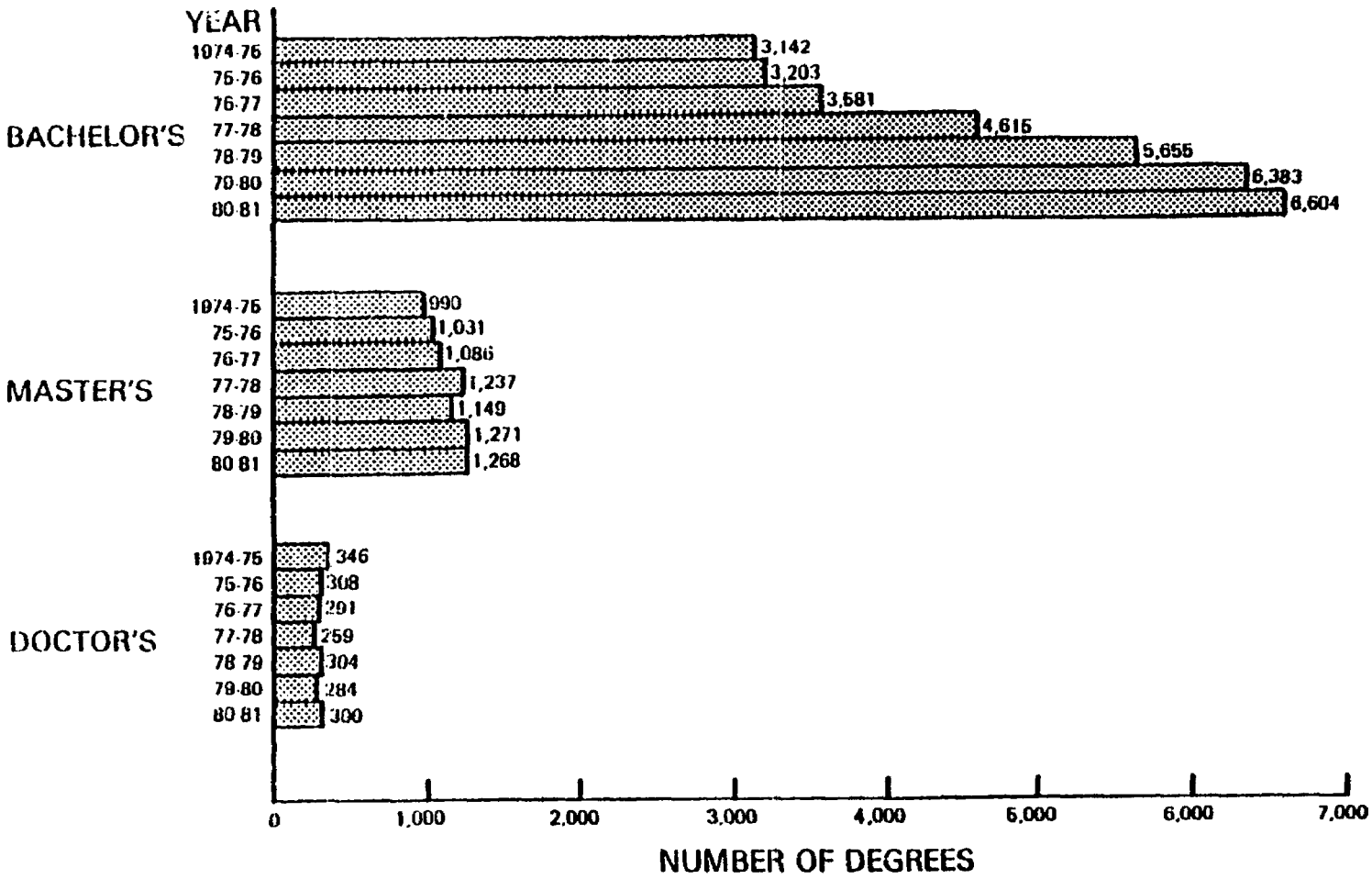


SOURCES: NATIONAL SCIENCE FOUNDATION
NATIONAL CENTER FOR EDUCATION STATISTICS

MCS 83-451
1-6-83

Figure 30

DEGREES AWARDED IN CHEMICAL ENGINEERING

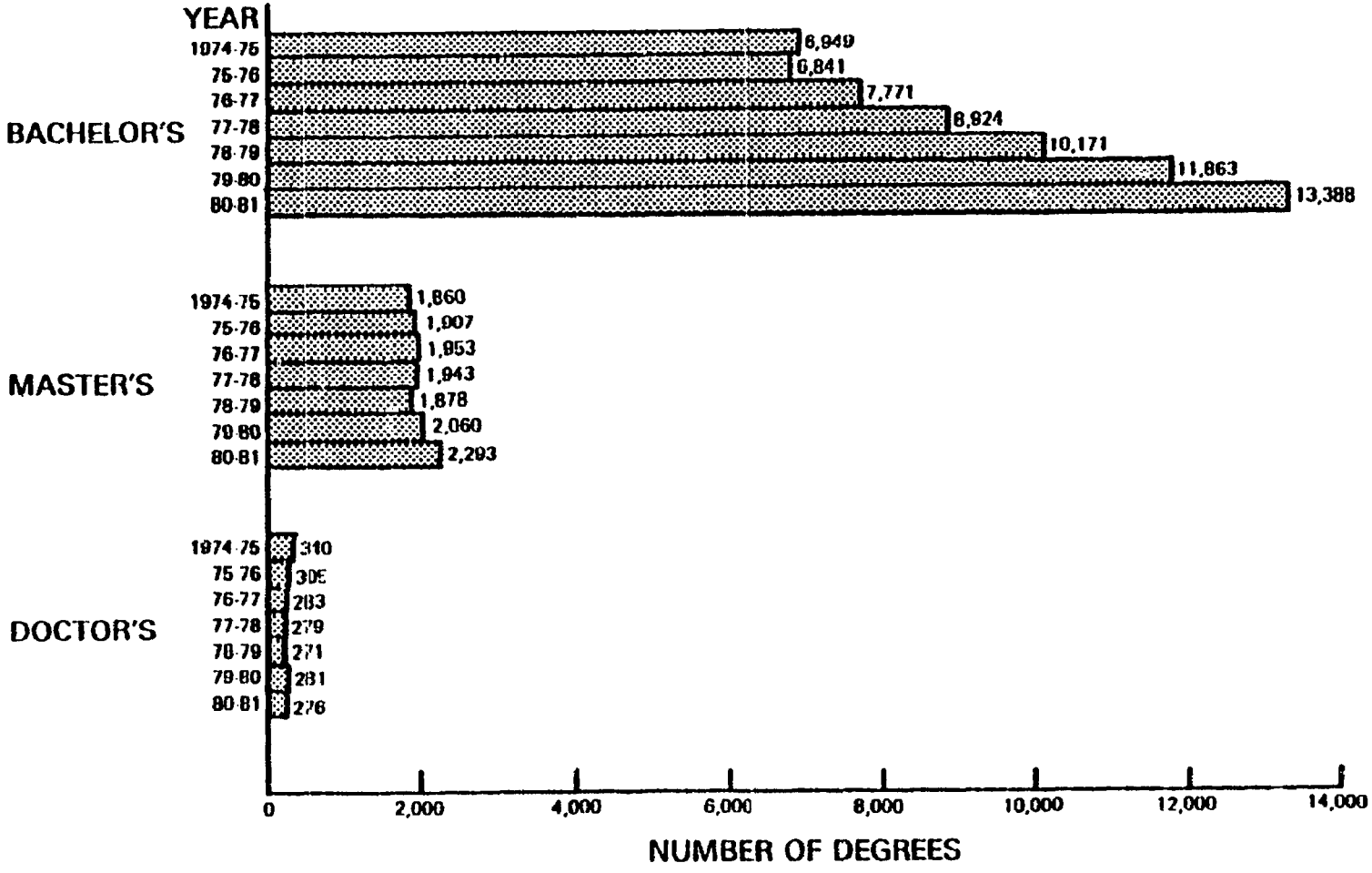


SOURCES: NATIONAL SCIENCE FOUNDATION
NATIONAL CENTER FOR EDUCATION STATISTICS

MCS 83-453
1-83

Figure 31

DEGREES AWARDED IN MECHANICAL ENGINEERING

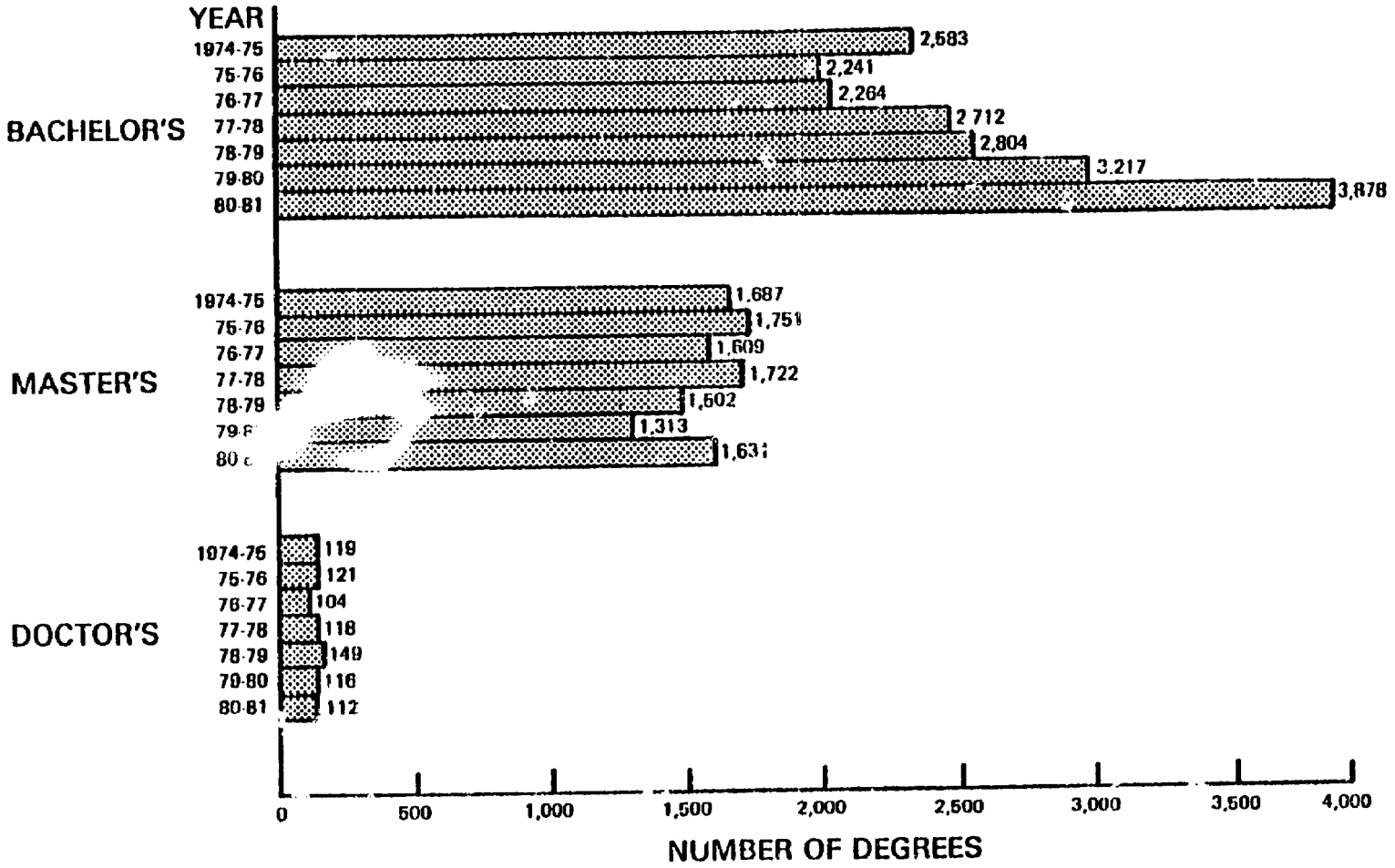


SOURCES: NATIONAL SCIENCE FOUNDATION
NATIONAL CENTER FOR EDUCATION STATISTICS

MCS 83 449
1-83

Figure 32

DEGREES AWARDED IN INDUSTRIAL AND MANAGEMENT ENGINEERING

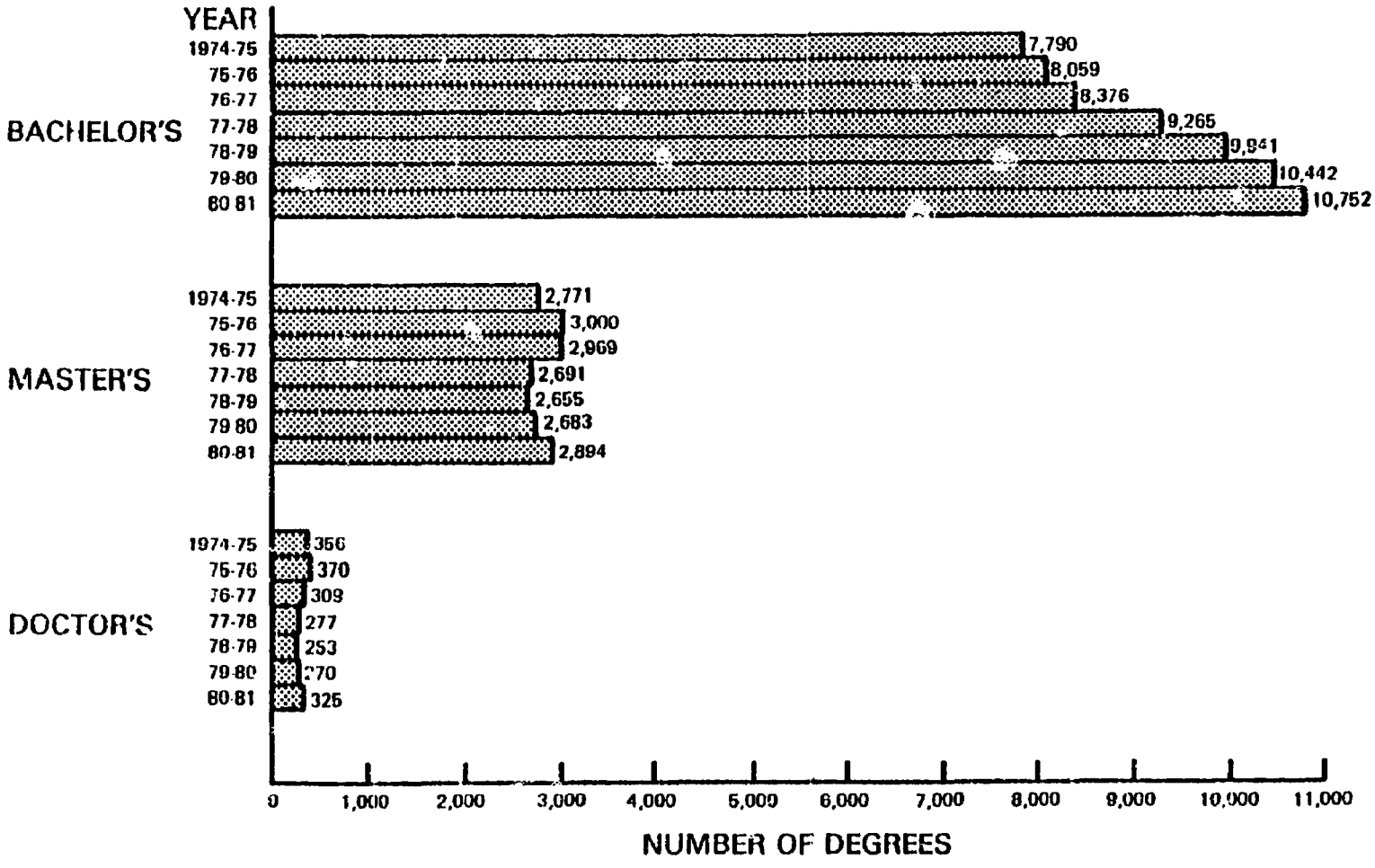


SOURCES: NATIONAL SCIENCE FOUNDATION
NATIONAL CENTER FOR EDUCATION STATISTICS

MCS 83-450
1-5-83

Figure 33

DEGREES AWARDED IN CIVIL, CONSTRUCTION, AND TRANSPORTATION ENGINEERING

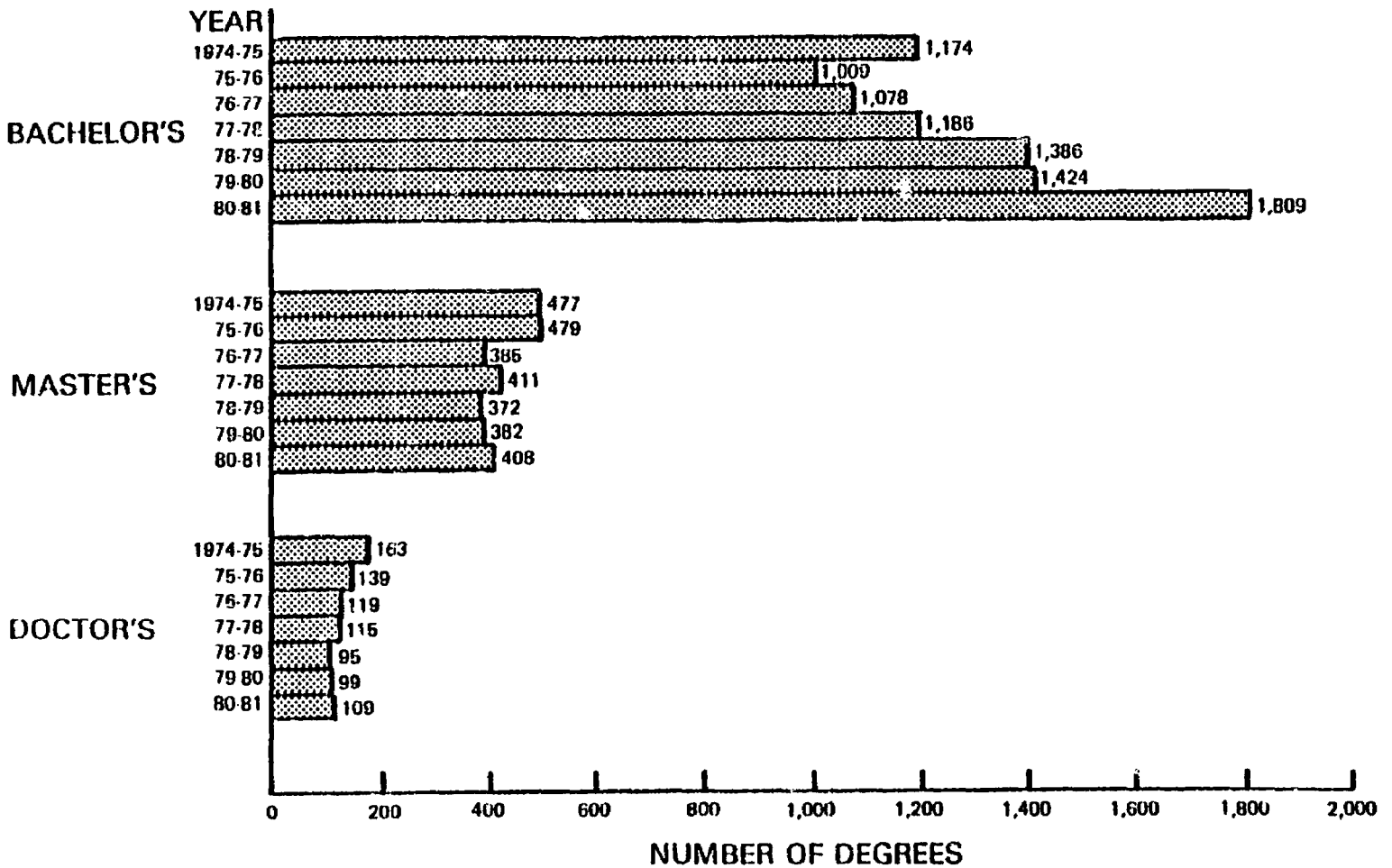


SOURCES: NATIONAL SCIENCE FOUNDATION
NATIONAL CENTER FOR EDUCATION STATISTICS

MCS 83-452
1-5-83

Figure 34

DEGREES AWARDED IN AERONAUTICAL/ASTRONAUTICAL ENGINEERING

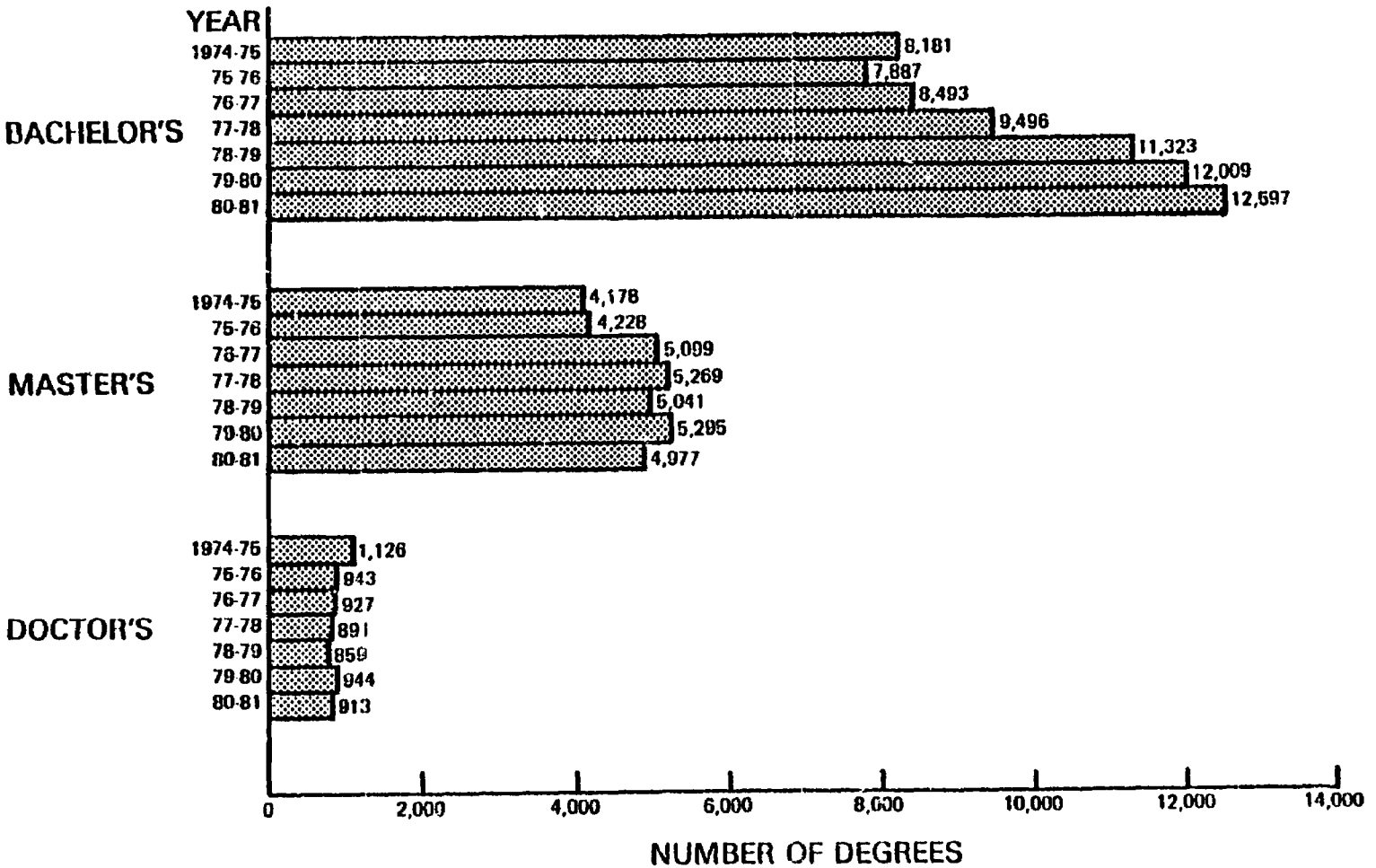


SOURCES: NATIONAL SCIENCE FOUNDATION
NATIONAL CENTER FOR EDUCATION STATISTICS

MCS 83-454
1-83

Figure 35

DEGREES AWARDED IN OTHER ENGINEERING

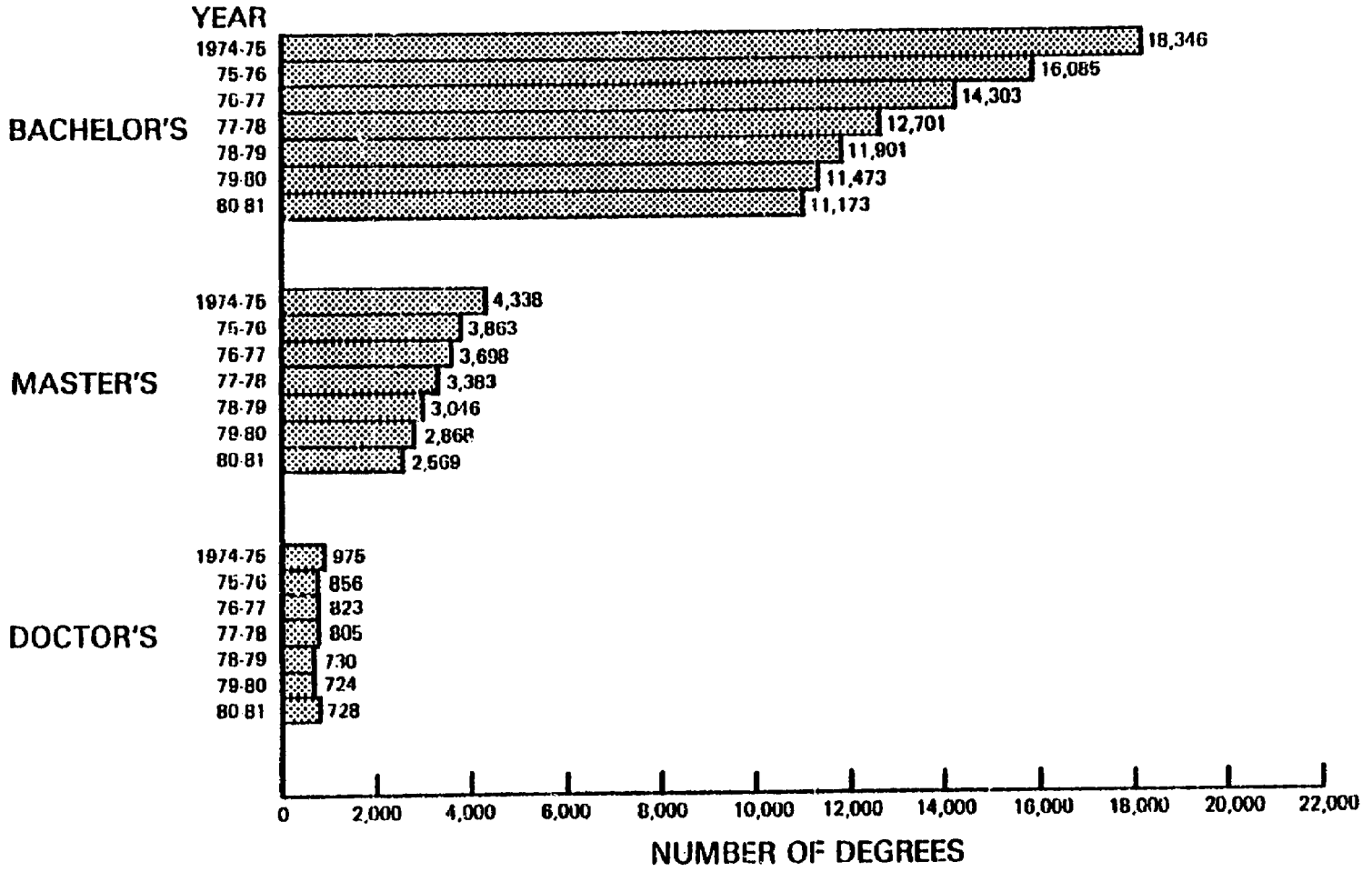


SOURCES: NATIONAL SCIENCE FOUNDATION
NATIONAL CENTER FOR EDUCATION STATISTICS

MCS 83-448
1-5-83

Figure 36

DEGREES AWARDED IN MATHEMATICS

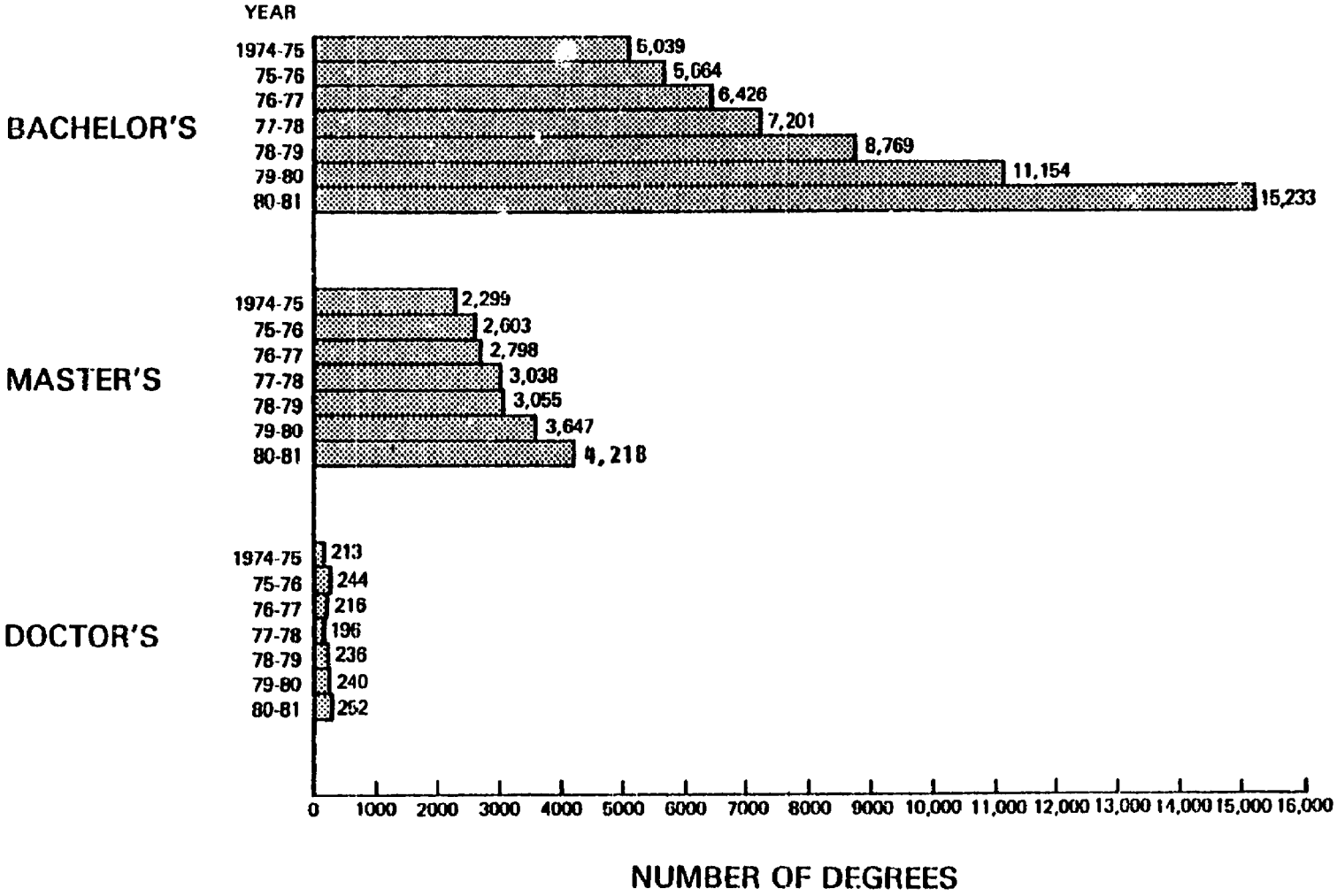


SOURCES: NATIONAL SCIENCE FOUNDATION
NATIONAL CENTER FOR EDUCATION STATISTICS

Figure 37

MCS 83-461
1-5-83

DEGREES AWARDED IN COMPUTER SCIENCE

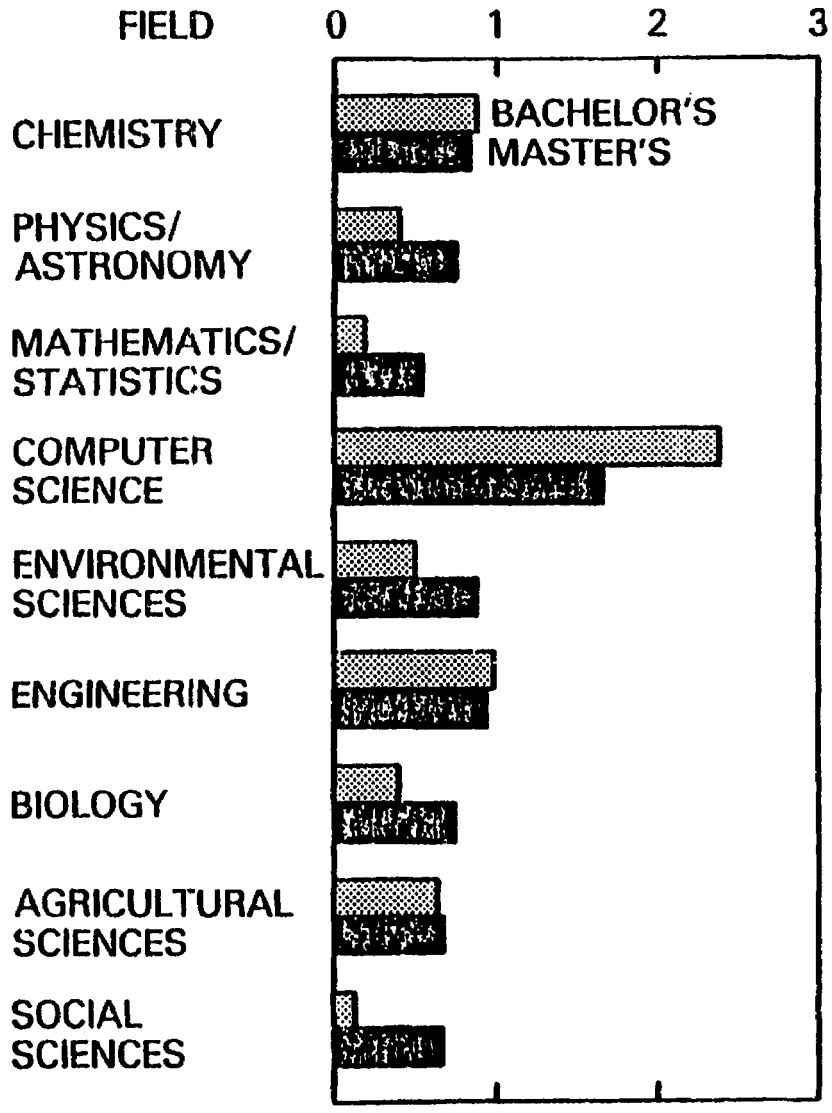


SOURCES: NATIONAL SCIENCE FOUNDATION
NATIONAL CENTER FOR EDUACATION STATISTICS

Figure 38

MCS 83 464
1-5 87

RATIO OF 1978 SCIENCE/ ENGINEERING DEGREE RECIPIENTS[a] EMPLOYED IN FIELD RELATIVE TO GRADUATES IN FIELD WHO ENTERED THE LABOR FORCE: 1980



[a]EXCLUDES INDIVIDUALS ENROLLED FULL TIME IN GRADUATE SCHOOL.

SOURCE: NATIONAL SCIENCE FOUNDATION

MCS 81-1967
REV. 1-6-83

Figure 39

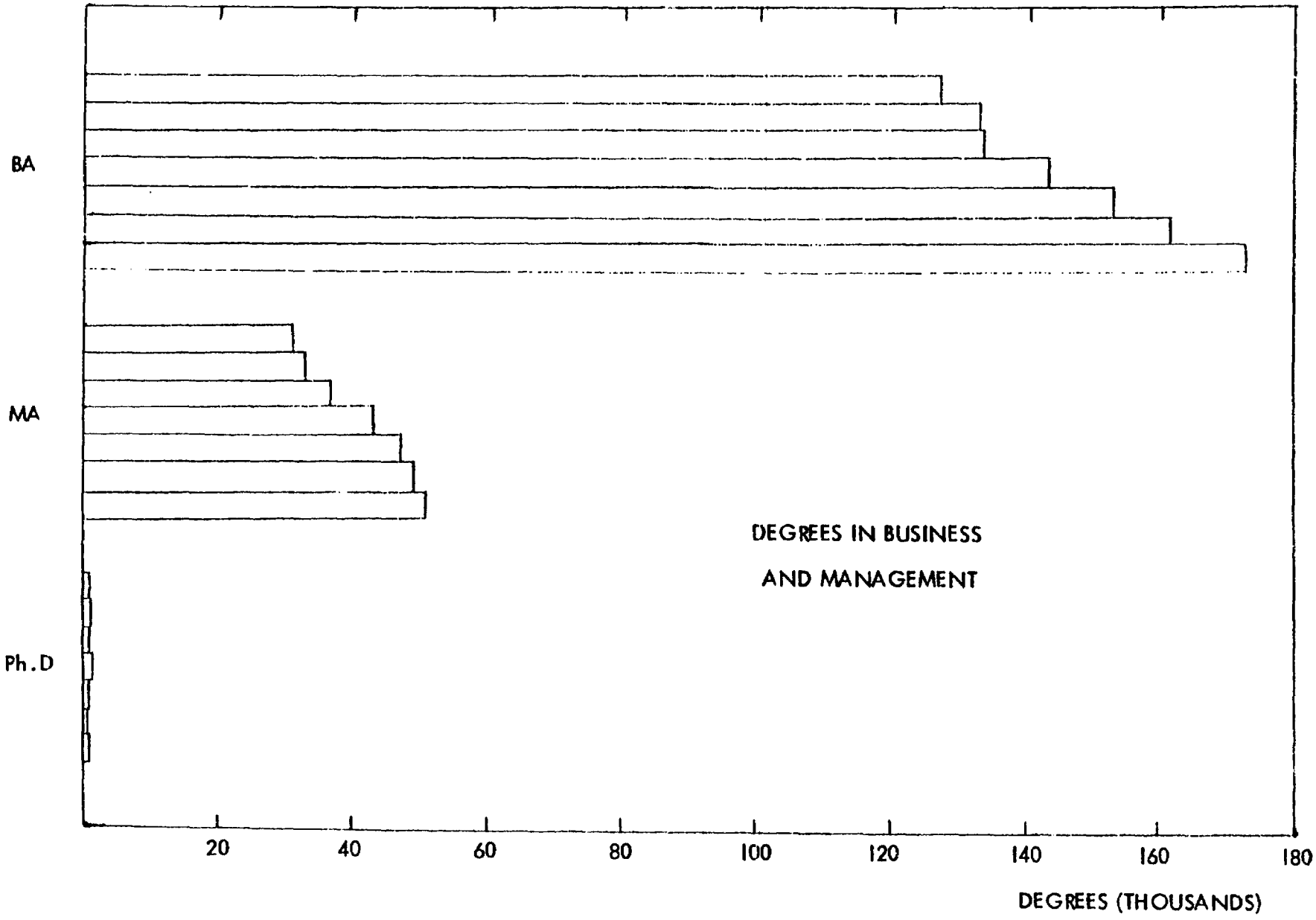

BA
MA
Ph.D
DEGREES IN BUSINESS
AND MANAGEMENT
20
40
60
80
100
120
140
160
180
DEGREES (THOUSANDS)

Figure 40

ARTIFICIAL INTELLIGENCE

An Assessment of the State-of-the-Art and
Recommendations for Future Directions

Prepared for
NSF Information Technology Workshop
Xerox International Center
Leesburg, Virginia

January 5-7, 1983

by

David Waltz, Chairman	University of Illinois, Urbana
Michael Genesreth	Stanford University
Peter Hart	Fairchild Advanced Research and Development Lab
Juris Hartmanis	Cornell University
Gary G. Hendrix	Symantec, Inc.
Aravind K. Joshi	University of Pennsylvania
John McDermott	Carnegie-Mellon University
Thomas Mitchell	Rutgers University
Joel Moses	Massachusetts Institute of Technology
Nils Nilsson	SRI International
Robert Wilensky	University of California, Berkeley
William A. Woods	Bolt, Beranek & Newman, Inc.

ARTIFICIAL INTELLIGENCE

Abstract

This report covers two main AI areas: natural language processing and expert systems. The discussion of each area includes an assessment of the state-of-the-art, an enumeration of problem areas, opportunities, recommendations for the next 5-10 years, and an assessment of the required resources. Also included is a discussion of possible university-industry-government cooperative efforts.

TABLE OF CONTENTS

SUMMARY

Artificial intelligence (AI) comprises both applied and basic research. Existing applications include programmed systems that allow natural language (NL) access to databases, and "expert systems" that can perform tasks in such areas as medical diagnosis, mineral exploration, and the configuration of computer systems. A number of applications are now available as commercial products. Recently, there have been reports in the popular press about AI, and great expectations for a wide variety of intelligent application systems have been aroused in the public, industry, government, and the military. AI is also central to the Japanese Fifth Generation Computer effort, and many sources have stressed the importance to the US of maintaining leadership in intelligent systems. It is important, however, not to expect too much from AI too soon. Many of the important applications we can imagine require substantial progress in basic research.

As in many other areas of computer science and engineering, such progress is now impeded by an extreme shortage of manpower and equipment, both at universities and industrial laboratories. Furthermore, the shortage of researchers is not likely to be corrected in the near future. A survey of the faculty at the top ten universities doing AI work has revealed that there are only about 16 natural language researchers, and only about 28 expert systems researchers. At best we can expect only 15-20 new Ph.D.s per year, nearly half of whom will go to industrial positions, based on recent data. Moreover, experience has shown that research progress in these areas depends strongly on having a "critical mass" of researchers; fewer than half of the top ten universities have critical research mass in either area. Better facilities are essential in order to retain faculty at universities, to attract new graduates to academia, and to maximize research progress.

The following are our recommendations for increasing our rate of research progress over the next ten years.

(1) Continue strong funding for basic research in AI, and avoid shifting too much emphasis to applications with short-term payoffs, no matter how attractive they seem; otherwise we will not be generating new application systems in ten years.

(2) Provide more and better equipment for universities and laboratories doing AI research.

(3) Encourage resource sharing by funding specific groups to supply and maintain a common body of research tools, e.g., software systems for knowledge representation.

(4) Encourage development efforts in military and industrial labs, and increased contact between such groups and basic researchers in universities and research laboratories. Industries could be encouraged to donate equipment and research funds to universities, e.g., by allowing tax credits for unrestricted research funding.

(5) Encourage industrial research laboratories to help by advising Ph.D. research wherever possible.

(6) Maximize faculty research and research supervision time by providing academic year salaries as well as summer salaries.

(7) Institute three-four-year research initiation funding, including equipment funds, for promising new graduates who agree to stay at universities.

(8) Encourage the design of novel supercomputer

architectures that take AI needs into account; current supercomputers are designed for numeric computations that are of little use or interest to AI, though AI badly needs computers with greater power.

(9) Encourage researchers in adjacent fields to develop AI knowledge and skills, and to participate in novel joint projects, especially in the area of cognitive science.

1.0 *INTRODUCTION*

1.1 *What is AI?*

Artificial intelligence (AI) is the attempt to (1) understand the nature of intelligence and (2) produce new classes of intelligent machines through programming computers to perform tasks that require reasoning and perception. The goal of AI as a whole is to produce machines that act intelligently. By "act intelligently," we mean to cover a broad range of activities, only some of which are directly human-like; our machines may eventually be far better at certain intelligent tasks than people are (much as a calculator does long division better than people do), yet may lack other human characteristics (e.g., ambition, fear, general world knowledge, mobility). Important characteristics that such machines require before AI could be said to have succeeded include common sense; the ability to learn from experience; the ability to accept, generate and act appropriately on natural language input; perception; and general situation assessment.

1.2 *The Coverage of This Report*

For the purposes of this report, we concentrate on natural language (NL) processing and expert systems.[1] Both of these areas have near-term practical potential, and yet are sufficiently open-ended to remain interesting to researchers past the next ten years.[1]

The biggest AI news of the recent past has been the commercial introduction and industrial use of a number of AI systems, especially NL and expert systems. This news is significant because (1) it has quieted critics who argued that AI would never produce useful results, and (2) the applications themselves have high intrinsic value. Some specific systems include:

INTELLECT (AI Corp., Waltham, MA). A natural language system that can be added to a customer's existing data base.

STRAIGHT TALK (Symantec for Dictaphone Corp.). A natural language data base system embedded in a microprocessor-based word processor, intended for uses such as the storing of address, phone number, organization, and salary information.

RI (Designed at Carnegie-Mellon University for Digital Equipment Corp.). An expert system that can check and correct the configuration of the 450 or so components that can go into a VAX computer.

PROSPECTOR (SRI International). An expert geologist system that can find commercially exploitable mineral deposits from assay data.

The Machine Intelligence Corp. Vision module (based on research originated at SRI), a system that can be "taught" by a non-programmer to recognize and act upon a variety of parts on an assembly line, when used with a robot arm.

DENDRAL (Stanford). An expert system for discovering the molecular structure of organic compounds from mass spectrogram data.

MACSYMA (MIT). A general mathematical aids system, including the abilities to do symbolic integration, factoring, simplification, plotting, and much more.

MOLGEN (Stanford). An expert system for designing experiments in molecular biology. In addition, there are a number of systems that, while not being marketed, have received fairly wide use, for instance:

MYCIN (Stanford). An expert system for medical diagnosis and treatment prescription.

EMYCIN (Stanford). MYCIN with specific medical knowledge removed, EMYCIN has been used by non-programmers to make expert systems in areas other than medical, such as tax advising and estate planning.

INDUCE/PLANT (University of Illinois). A system that has learned to generate its own rules for diagnosing soybean diseases from examples presented to it. It has better performance than that of most experts.

Dipmeter advisor (Schlumberger-Doll). A system for interpreting oil-well log data.

LISP machines (BBN; LMI, Inc.; MIT; Symbolics, Inc.; Xerox). Computers specialized for the LISP language that pioneered the ideas of the personal workstation and user-friendly programming environments.

There are, in addition, a number of companies that plan to introduce or use internally AI products in the near future, including NL systems from Texas Instruments, Cognitive Systems, Inc., Symantec, and Hewlett-Packard, and expert systems from Teknowledge, Fairchild, Intelligenetics, Schlumberger, and others.

1.3 *Overview of This Report*

In the sections that follow, we first discuss in some detail the nature of the most important current technical and scientific issues in natural language processing and in expert systems. We then discuss organizational problems, including difficulties involving manpower, equipment, funding, education, and cooperation between industry, government, and universities. This is followed by a list of recommendations that we believe can aid in maintaining the U.S. lead in these important areas.

2.0 *NATURAL LANGUAGE RESEARCH*

2.1 *Introduction*

Endowing computers with an ability to communicate with humans in natural language (e.g., ordinary English) has been a major topic of research in AI for over 20 years. The ultimate goal of creating machines that can interact in a facile manner with humans remains far off, awaiting breakthroughs in basic research, improved information

1 Because we have researchers primarily from these two areas, in turn because these two areas were selected for special consideration by the workshop steering committee. However, there are many other important problems in AI that are not well-represented here. The areas of computer perception and pattern analysis are particularly unfortunate omissions. Other areas of major AI research are included to some degree, because they overlap other sections of this report (robotics, parallel and distributed hardware, software and programming environments and information sciences). In each of these areas, however, AI has specific needs, often somewhat different from the rest of computer science, which deserve consideration. We hope that these important AI research areas will be considered in future assessment and planning meetings, especially because progress in natural language and expert systems ultimately depends critically on virtually all other AI areas.

processing algorithms, and perhaps alternative computer architectures. However, the significant progress experienced in the last decade demonstrates the feasibility of dealing with natural language in restricted contexts, employing today's computers.

Continuing research in this area seems likely to lead both to progressively more practical, cost-effective systems and to a deeper understanding of the natural phenomena of NL communication. Each of these goals has importance in isolation; pursuing them simultaneously enables progress on each to support progress towards the other.

2.2 *Application Areas*

Natural language processing has a broad range of possible application areas.

Machine Translation or MT involves using machines to convert documents written in one natural language to corresponding documents written in another language, but with equivalent meaning. MT was proposed in 1946 and became a forerunner of today's work in AI. Creation of fluent translations remains an elusive goal.

Document understanding involves reading documents by machine, and assimilating the information the documents contain into a larger framework of knowledge. After reading a document a device of this sort might produce an abstract of it, alert people who are likely to be interested in it, or answer specific questions based on the information it contains. If such a device were to read and assimilate many documents, it might act as a librarian, directing users to especially pertinent references.

Document preparation aids could perform the task of an experienced editor, detecting errors in spelling and grammar, and suggesting ways to rephrase passages of text to make them more understandable and to make them conform to patterns of high-quality language usage.

Document generation, a task related to document understanding, involves translating information stored in a formal language in a computer's memory into ordinary language. The documents produced might be single sentences or extensive texts. For example, information encoded in a formal language regarding the repair of an electro-mechanical device could be used as the basis for mechanically generating instruction manuals in a variety of natural languages. Moreover, from the same formal description, different manuals could be generated for different audiences such as end users, repair personnel, and engineers. Ultimately, documents could be tailored to the background of each particular individual, making each document more understandable and generating the appropriate level of detail.

Systems control, the applications area with the greatest promise for near-term achievement, involves the use of NL in the control of computer systems. By coupling a natural language interface with different types of devices, a range of possible systems may be produced, including systems that (1) provide answers to questions by accessing large data bases; (2) control such complex systems as industrial robots, power generators, or missile systems; (3) furnish expert advice about medical problems, mechanical repairs, mineral exploration, the design of genetic experiments, or investment analysis; (4) create graphical displays; (5) teach courses in a broad range of subjects, interacting with students in English.

Practical systems for (1) above are already commercially available, and rapid development of this area can be expected over the next several years. Little work has been done on area (2), but there appears to be no special technical obstacles to producing elementary systems in this area if programs are undertaken. Area (3) involves integrating work on natural language processing with work on expert systems; some limited demonstration programs using stylized input and canned output have been produced already, but much remains to be done. Areas (4) and (5) require considerable new work, but significant progress could be made if a concentrated effort were undertaken. (Note: Work in this area could help educate people in computer science in general and in AI in particular.)

In considering natural language as the command language for controlling computer systems, it is important to keep in mind that English is well-suited to some kinds of man-machine interaction and poorly suited for others (such as those involving extensive manipulation of numbers). English is appropriate

- when dealing with computer-naive users
- for tasks which computer experts do infrequently
- in situations where English is more concise than formal language
- for activities in which natural language is the subject of analysis, e.g., intelligence gathering.

Speech understanding involves the coupling of natural language processing with acoustic and phonetic processing to achieve a device that can understand spoken as opposed to typed input. Advances in this area are currently inhibited primarily by the lack of satisfactory acoustic/phonetic devices for recognizing sequences of individual words in continuous speech. Special purpose parallel hardware for word recognition is being pursued by researchers outside AI.

2.3 *Importance and Economic Impact*

Perhaps the most important economic factor of the current age is that, as a society, we are moving away from an economy based on the manufacture and dissemination of goods to an economy based on the generation and dissemination of information and knowledge. Much of the information and knowledge is expressible in English and much of the task of gathering, manipulating, multiplying and disseminating it can be greatly aided by computers. Thus research in NL understanding can have a two-fold positive impact in our shifting economy:

(1) NL can enable computers to interact with users in ordinary language, and therefore it can make computer power available to segments of the population that are unable or unwilling to learn one of the formal languages usually required for interaction with computers.

(2) NL can increase knowledge productivity by providing mechanical means for manipulating knowledge expressed as natural language text.

2.4 *State-of-The-Art in Natural Language Processing*

Currently, we understand how to do a reasonably good job of literal interpretation of English sentences in static contexts and limited, well-structured domains of application. This is not to say that there are no open problems in this area, but rather, compared with the other aspects of language to be discussed, there is a substantial body of

known results and proven techniques.

A number of good application areas are now possible. Examples include NL database front ends; NL interfaces for expert systems, operating systems, system HELP facilities, library search systems, and other software packages; text filters, text summarizers; machine-aided translation; and grammar checkers and critics.

Parsing algorithms for syntactic analysis of sentences and techniques for semantic interpretation (to determine literal meaning) are well developed, making possible many practical applications.

It is now possible to think about a much wider range of types of NL processing, because machines have become substantially larger and address spaces have become reasonable for NL systems. In the past, a great deal of effort was devoted to attempts to collapse code length so that space could be made available to expand the capabilities of NL programs. With the advent of LISP machines, this is no longer such a problem, and we can trade off space for ease of programming.

However, extensive resources are required to develop a new NL application even in areas we understand quite well, because there is no such thing as a small natural language system. If a system is to accept natural language, that is, unrestricted text or what people naturally think of saying in the manner they think of saying it, it must have a large vocabulary, a wide range of linguistic constructions, and a wide range of meaning representations. A small system cannot be very natural.

Existing techniques begin to break down when we begin to scale up to more open-ended applications where inherent limitations of the domain are no longer adequate to resolve ambiguities in the language, or where sophisticated discussion of time or three-dimensional space is required, or where discussion and modelling of the beliefs, goals and rational behavior of intelligent agents is required.

2.5 *Research Areas in NL*

The most active current areas of research (and the most promising for new breakthroughs) lie in the area of recognizing the intent of speakers and the relationships between sentences in continuous discourse, taking into account the structure of the preceding discourse, the non-linguistic aspects of the situation of utterance, and models of the beliefs and goals of the communication agents.

Historically, our understanding of the phenomena of language has moved gradually from the most visible and salient aspects such a phonology, lexicon, and syntax towards the more internal and intellectual elements of semantics, pragmatics, and reasoning. Like the stages of Piagetian development it has seemed that we must first gain a mastery of the earlier stages before proceeding to explore the later ones.

2.5.1 *Plan Recognition*

A recent innovation, with enormous potential for increasing our understanding and capabilities in these areas, has been the evolution of a methodology in which communication is treated as a special case of a goal-oriented action in a common framework with non-linguistic actions. This allows for planning and reasoning about goal-oriented behavior which involve communication and acquisition of information as a part of the overall activity of a system. As a simple example, in order to get into a locked room occupied by a human, a robot might construct and execute a plan that would lead to its saying, "Please unlock the door so I can come in."

2.5.2 *Speaker's Intent and Plan Recognition*

The planning approach to the problem of communication has put new flesh on the basic skeleton of the theory of speech acts advanced by philosophers and promises significant advances in linguistic fluency of communicating machines. It also provides a framework for non-linguistic communication through actions such as pointing or displaying a picture. There are, however, substantial technical problems that must be addressed in order to realize the promise of this approach. These include development of reasoning systems capable of modelling and reasoning about the beliefs, goals, and actions of rational agents; the representation, organization and access of the knowledge necessary to support such reasoning; and the discovery of frameworks, methods and algorithms capable of combining syntactic, semantic, pragmatic and general world knowledge to perform the overall task of understanding an utterance in context.

2.5.3 *Representation and Common Sense Reasoning*

The emerging focus on problems of interpretation in context have suddenly put great stress on the problems of knowledge representation and common sense reasoning. The solution of these problems requires extensive use of general world knowledge and knowledge about the rational behavior of intelligent, communicating agents. It is impossible to overemphasize the importance of representation and reasoning for this new stage of learning about language.[2] To extend existing systems we must address many fundamental problems.

(1) We must decide exactly what knowledge to include in an NL system. This type of knowledge is considered by people to be "common sense," and has never been codified.

(2) We must discern classes of objects, relationships and events that provide a national collection of terms which we may use to describe knowledge about the world, and we must describe their mutual interdependencies.

(3) We must devise appropriate notational systems.

(4) Knowledge must be organized in memory to facilitate access. In particular, facts may need to be recalled upon demand, and facts relevant to the current context may need to be inferred automatically.

(5) Knowledge and reasoning about the physical world is required for determining the referents of noun phrases, interpreting prepositional phrases, disambiguating word senses and generally understanding text that refers to some physical situations. This kind of knowledge is also critically important for judging whether an utterance is literally plausible, as opposed to metaphorical, humorous, sarcastic, or in error.

2.5.4 *Generation*

The areas described so far have treated language as an input modality only, i.e., they have concentrated on language understanding. Current NL systems do not often use NL as an output modality, but this situation is expected to change quite radically in the future. Systems of the future must deal with situations where the system understands that a user doubts, is confused by, or does not understand the system's response. Such systems must be prepared to paraphrase or explain their responses.[2]

2 The research areas of knowledge representation and common sense reasoning are areas likely to have considerable impact on expert sys-

There are three main aspects of generation: (1) deciding what to say, (2) deciding how to say it, and (3) finally saying it. Thus, it should be clear that the previously mentioned research areas (especially planning, knowledge representation, and common sense reasoning) are highly relevant to generation as well as to understanding.

2.5.5 *Algorithms*

The study of the formal properties of algorithms needed to manipulate the various types of NL representations mentioned above, is itself an important research activity. We know something about parsing algorithms and algorithms for certain logic systems. We will need to know much more about algorithms, for example about those that allow us to combine syntactic, semantic, and pragmatic knowledge. One recent example is the discovery that grammatical representations that combine syntax and semantics (i.e., provide grammatical representations that are convenient for semantic interpretation) have properties which make their parsing only slighly more complex than the parsing of context-free grammars.

2.5.6 *Scaling Up -- Learning*

If a NLP system is to be useful, it must be able to handle a large vocabulary and have access to knowledge bases. This imposes two constraints upon the design of an NL system. First, it must operate efficiently when it is in possession of relatively large amounts of knowledge. Second, means must be found for building up large knowledge bases appropriate for NL understanding and for progressively expanding a system.

Most working AI systems are of limited scope, and so there has been little actual experience with truly large-scale AI systems. Therefore, it is an act of faith to assume that our current techniques will be effective in vastly larger systems. One possible avenue for research involves the further investigation of techniques from data base management, in which problems of scale are routine, but in which the data has a much less complex structure. Techniques specific to the management of large collections of complex knowledge need to be developed.

The desirability of a large and incrementally expandable system poses some quite general questions of system design. In addition, some problems arise that are singular to NLP. For example, it would be desirably to build a system that could learn both vocabulary and world knowledge by dialogue in NL with a user, or by reading text (e.g., dictionaries, encyclopedias, texts, stories, etc.). The simpler aspects of this problem are within our current understanding. However, at the extreme end, one encounters problems of machine learning that require fundamental advances in our basic understanding.

2.6 *Opportunities in Natural Language*

Having reviewed the state-of-the-art and the major research problems for the next ten years, we would like in this section to briefly state some of the short-term goals that could lead to significant improvements in NL technology. The highest priority should be given to the basic research issues enumerated in Section 2.5 above, to insure new application areas in ten years.

Although the research areas described above will require ten years or longer to arrive at general solutions, it is possible to make incremental contributions toward the limited handling of various discourse phenomena, such as ellipses (i.e., the omission of words that can be understood in context), recognizing a user's intent in restricted situations, discovering some of the user's beliefs from the presuppositions of the user's input, modeling aspects of common sense, learning by reading text or engaging in dialogue, etc. It should be emphasized, however, that one should not expect progress in the basic research topics to be strictly incremental. Major breakthroughs are required in order to obtain general and principled solutions for these topics.

Research and development of good NL tools should pay excellent dividends. We now have available a few parsers and knowledge representation systems, which should make it possible to build new systems more rapidly by using off-the-shelf components for programs. Better documentation is needed for these systems, and we still need more experience in tailoring systems from these components, but there is the opportunity to begin building custom NL systems now, for limited task domains.

Because it is now possible relatively easily to produce special-purpose chips, the identification of potential parallelism in NL processing has gained importance. We can realistically consider algorithms for highly parallel word sense selection, truly concurrent syntactic, semantic and pragmatic evaluation of sentences, and speech format extraction, with the expectation that such work can lead to novel machine architectures more appropriate for NL processing.

Many good novel applications are possible within the next ten years, provided that we can solve some of the basic research problems enumerated above. Many of these arise in conjunction with "information utilities" (i.e., information services available via phone or cable connections). Possible public services include automatic directories of names, addresses, yellow pages, etc.; electronic mail; on-line catalogues and ordering facilities; banking and tax services; routing directions (like AAA); access to books and periodicals through titles, authors, topics, or contents; and undoubtedly many others. All of these services could also have on-line NL help facilities and manuals. There are parallel needs in business and in the military services, e.g., for command and control, inventory control, ordering and shipping, coordination of organization planning and plan execution, and so on.

Other good application bets concern the control of systems via NL. For example, we should, within ten years, be able to solve the problem of instructing robots in NL, much as one would instruct a human assistant. This can allow the robot to understand the goals of the instruction, rather than just the means for achieving the goals, as is the case now with teaching-by-leading-through-motions or "teaching" by ordinary programming. Understanding goals would help a robot in dealing with a wider range of situations and in recovering from errors.

Research on speech understanding should have a high priority. Once continuous speech understanding is

---

tems work as well. Current expert systems are based on representations chosen specifically for an application, with little general scientific understanding of the power or limitations of those representations or the consequences which may emerge at later stages of development from representational decisions made at the outset. There is already a recognized need for all expert systems to be able to explain their reasoning and conclusions (see below).

possible, the number of good NL application areas will grow dramatically. Continuous speech systems seem close enough to reality (commercially available under $20,000 within 10 years, provided that sufficient research funds are available) so that we should begin to think more seriously now about the realm of new application areas that speech systems will open up. It is critically important that we at the same time make substantial progress on the fundamental problems in NL understanding, so that we can move rapidly to produce useful systems. Speech recognition alone, without NL understanding and action components, would be of very limited value.

3.0 *EXPERT SYSTEMS*

Expert systems research is concerned with the construction of high-performance programs in complex domains. By "high performance" we mean functionality and efficiency comparable with, or better than the best human experts. By "complex domains" we mean those application areas requiring substantial bodies of knowledge, often of an uncertain or judgmental nature. This potential for capturing judgmental knowledge expands the range of problems to which computers have been and potentially could be applied.

What differentiates the expert system methodology from traditional computer programming is an emphasis on the symbolic manipulation capabilities of computers, in particular the declarative representation of world knowledge, the explicit encoding of heuristics, and the exploitation of non-numeric data structures for computer simulations.

3.1 *State-of-The-Art*

As just mentioned, expert systems differ from more conventional computer programs (which, like econometric models, for example, may possess considerable "knowledge") by their ability to deal with uncertain and judgmental knowledge, represented in a symbolic (often declarative) form. Expert systems differ from other AI programs, such as natural language understanding programs, because of their concern for subjects that usually require specialized training in a professional field, such as medicine, law, mathematics, or computer circuit design. It is perhaps one of the most surprising developments of the past ten years that a useful portion of the expert knowledge required for high-level performance in many of these fields can in fact be encoded in computer programs.

In addition to high performance, most of these systems can explain their conclusions so that their users have a better understanding on which to base action and greater confidence in the quality of the systems' decisions. This feature can best be explained by considering two contrasting approaches that could be used in constructing a program for medical diagnosis. In a system based on statistical decision theory using joint frequency distributions over symptoms and diseases, for example, the physician user might be informed only that, given the symptoms, the most likely disease is diabetes. A physician would have to take the conclusions on faith, and even if it were highly accurate, a user would be reluctant to depend blindly on such a system. This use of unexplained conclusions is in sharp contrast to what most expert systems can do. Current systems can describe the "chain of reasoning" employed by the system in reaching its conclusions. This chain refers to the judgments, assumptions, rules and intermediate conclusions used by the system. Physicians find these kind of explanations absolutely essential when deciding whether to rely on the machine's diagnosis or not.

Despite their success, current expert systems suffer from a variety of limitations. Among those shortcomings that ought to yield to concentrated research efforts during the next ten years or so are the following: overly narrow domains of expertise; inadequate communication channels with the user (e.g., need for better natural language and graphics); inability to represent certain kinds of knowledge easily (e.g., knowledge about processes, time, three-dimensional space, beliefs of the user); and the great difficulty of building and modifying the expert knowledge bases on which these systems are based.

It is this last area, knowledge acquisition, where we can expect the most difficulty. Current systems are built by having computer scientists interview experts in the domain of application. The knowledge obtained from these experts, usually in the form of English sentences, must then be structured by the computer scientist (often called a "knowledge engineer") so that it can be unambiguously and economically represented in the computer. (Incidentally, this process of precisely structuring knowledge, for instance, geological knowledge, for the computer can just as well be thought of as an advance in the science of geology as an exercise in system engineering.) Although it is reasonable to expect that we will be able to develop sophisticated computer aids for the knowledge acquisition process, it must be remembered that to completely automate the knowledge acquisition process will require rather dramatic advances in other areas of AI such as natural language understanding and generation, and machine learning.

Other problems that will require many years of work involve connecting expert systems to complex perceptual channels such as vision and speech. Although some work has already been done in this area, the general problems of interpreting visual images and connected speech are difficult long-term research areas.

3.2 *Current Research Problem Areas*

Though we have reached a point where we can develop expert systems that can provide significant assistance within a narrow task domain, the systems which have been developed to date all suffer from several serious weaknesses. In general, there are (1) system development limitations, (2) competence limitations, and (3) use limitations.

System development limitations:

(a) Constructing an expert system requires several man-years of effort from a programmer with a background in artificial intelligence; since very few people have such a background, the number of expert systems currently being developed is small.

(b) The expert systems which have been developed to date are not at all general; each system is a "single-customer", system.

(c) Since the knowledge which an expert system has is collected over time, often from several experts, it is not uncommon for some of the system's knowledge to be inconsistent, there are as yet no good methods for identifying such inconsistencies let alone repairing them.

Competence limitations.

(a) Since the knowledge expert systems have is relevant

to a narrow domain that is somewhat arbitrarily delimited, the systems sometimes make myopic judgments.

(b) The systems do not have the ability to check their conclusions for plausibility; thus they sometimes make incredibly naive recommendations.

(c) Since the knowledge the systems have is almost exclusively "surface" (empirical) knowledge, they are unable to infer missing knowledge from general principles; thus their behavior degrades badly when knowledge is missing.

Use limitations:

(a) Almost no expert systems have natural language understanding systems as front ends; as a consequence, users may find expert systems unnatural to use.

(b) Though nearly all expert systems can provide explanations of how they arrive at their conclusions, these explanations are often not very convincing because they are not tailored to individual users.

(c) Most expert systems take longer to perform a task than is required by human experts.

3.3 *Research Opportunities*

While recent progress in expert systems has led to a number of practical programs, and to a strong interest by industry in this area, expert systems technology is still at a very early stage of development. Because we are at such an early stage of development, the single most important research investment in this area is probably to fund basic AI research. This section points out several specific research opportunities likely to lead to increased capabilities and a broadened impact for expert systems in the coming decade.

3.3.1 *Knowledge Acquisition and Learning*

Given the well-recognized knowledge-acquisition bottleneck, one major opportunity for increasing the power and decreasing development costs of expert systems is in developing new methods for knowledge acquisition and learning.

A small amount of research is currently going on in this area, ranging from development of interactive aids for validating, examining, and debugging large rule bases, to more basic research on automated learning and discovery of heuristics. The former type of system (e.g., TERIESIAS, Davis; SEEK, Politakis) has already been shown to be useful in development of expert systems. Systems of the latter type are still in the basic research stage, and further progress in this direction could have a major impact on expert systems technology.

Examples of interactive aids for debugging a rule base include:

TEIRESIAS [Davis, 1976]	System aids user in isolating faulty rule in chain of inferences that leads to incorrect conclusion.
SEEK [Politakis, 1982]	System collects statistics on rule performance over a database of known correct patient diagnoses, to isolate incorrect rules, and suggest possible revisions.

Examples of systems that infer new rules from provided data include:

Meta-DENDRAL [Buchanan, 1978]	Infers rules that characterize behavior of molecules in a mass spectrogram, for use in DENDRAL system for chemical structure elucidation; infers these from given set of molecules and their known spectra.
INDUCE/PLANT [Michalski, 1980]	Infers rules that characterize plant diseases, given data of symptoms and known correct diagnoses. These rules yield expert performance comparable to that attained using rules provided by human experts.

Examples of systems that learn heuristics (e.g., *control* knowledge, as opposed to factual domain knowledge) include:

LEX [Mitchell, 1982]	Learn heuristics for selecting among alternative applicable rules, in solving symbolic integration problems, analyzing the solutions, proposing heuristics, then generating new practice problems, etc.
EURISKO [Lenat, 1982]	Discovers new circuit structures for "high-rise" VLSI circuits, and discovers heuristics for guiding its search for new circuitry structure. This is an interactive system, and has been used in additional domains, such as elementary number theory, and naval fleet design.

While systems such as those noted above have demonstrated the utility and feasibility of computer aids for knowledge acquisition, further progress can have a major impact on development costs for expert systems, as well as on the level of complexity of systems which can be constructed. Specific problem areas for near-term research on knowledge acquisition and learning include research of methods for interactively and automatically analyzing and validating an existing knowledge base, and for isolating errors.

Other promising directions in this area include:

(1) Research on methods for inferring new inference rules from problem- solving experience. For example, in an expert system for interactive problem-solving, those problem solving steps provided by the user constitute inference steps that the system might assimilate and generalize into rules of its own for subsequent use.

(2) Research on compiling "deep" knowledge into more efficient "shallow" inference rules. (This is in the context of expert systems that integrate multiple levels of knowledge of the problem domain.)

(3) Longer term basic research on machine learning, essential to progress on developing shorter-term knowledge acquisition aids. Such basic research issues include developing methods for: (a) extending representational vocabulary; (b) learning by autonomously generating, solving and analyzing practice problems; and (c) learning domain knowledge by reading text-books.

3.3.2 *Representation*

The representation of knowledge continues to be an area of fundamental significance to AI. By changing representations one can drastically affect the functionality, efficiency, and understandability (and therefore modificability) of expert systems.

One key subtopic in this regard is the question of *representational adequacy*, i.e., the question of whether there is any way to encode certain facts within a language. In the past, research in this direction had led to results such as extension to predicate calculus necessary to handle default knowledge. Representation facilities are another key topic. In the past such research has led to the development of convenient specialty "languages" based on frames and semantic nets. Perhaps most important is the actual representation of certain aspects of "naive physics" knowledge, representation of time, space, matter, and causality.

3.3.3 *Inference Methods*

Inference methods are crucial to the expert system methodology so that programs can apply facts in their knowledge bases to new situations. While substantial work has been done on inference, both by logicians and researchers in AI, certain forms of inference of particular significance require further study. Important topics here include reasoning by default, reasoning by analogy, synthetic reasoning (i.e., design), and especially planning and reasoning under certainty.

3.3.4 *Meta-Level Architecture*

Meta-level architecture is a new area for research that has considerable potential significance. The goal is the construction of programs that can explicitly reason about and control their own problem-solving activity.

The approach here is to view the problem of problem-solving control as an application area in its own light, just like geology or medicine. Control recommendations can be expressed in a "declarative" fashion and a program can reason about these recommendations in deciding what to do. In this way, one can build a system of multiple representation and inference methods, supply it with facts about its goals and methods, and allow it to decide which to use in a given situation. Many traditional knowledge representation techniques, such as defaults and procedural attachments, can easily be expressed as meta-level axioms. Key subproblems include the meta-level encoding of standard results from theoretical computer science, and the compilation of programs built with meta-level architecture.

3.3.5 *Research on User Interfaces*

As expert system applications grow into increasingly complex problem-solving areas, the importance of high-quality user interfaces will increase as well. Progress here involves issues typically associated with man-machine interfaces, such as the need for high-quality graphics, friendly interfaces, fast response time, etc. In addition, the nature of expert systems applications usually requires that a system be able to explain its problem solving behavior, how it reached its conclusions, what inference steps were involved, and whether the problem at hand is one for which it possesses appropriate expertise. For example, an expert system for medical diagnosis and therapy is much more acceptable if it can explain the reasoning behind its diagnosis and therapy recommendations. While current systems have capabilities for enumerating the inference steps involved in reaching conclusions, more sophisticated explanation facilities would greatly improve the acceptability and utility of these systems. One significant opportunity for improving user interfaces (especially for naive users) is to incorporate natural language interfaces into expert systems.

4.0 *ORGANIZATIONAL PROBLEMS AND NEEDS*

AI has a number of special difficulties because of the ambitious, open-ended nature of its enterprise, and because of the size of the research community compared to the size of the technical problems. In addition, AI suffers from many problems that are shared with the rest of science and engineering.

Even restricted NL and expert systems can be very large, and since the success of an approach cannot really be measured until a system embodying it is completed and evaluated, the design-test-redesign cycle tends to be long. The development of a full-scale system is also a problem. Let us suppost that we want a NL system with a 10,000 word vocabulary; while a portion of a lexicon for the NL system can be extracted from a dictionary, each definition also requires separate attention, since dictionaries do not contain all the information necessary. If a team approach to construction is used, the team must be small in order to avoid the types of problems encountered in the contruction of operating systems -- as a programming team gets larger, its members tend to spend more and more of their time talking to each other about standardization, intermodule communication, updates, etc., and produce fewer and fewer lines of code per person per day. On the other hand, if a programming team is very small, it will simply take a long time to produce the necessary code because of individual programming speed limitations. Similar problems arise with the building of knowledge bases for expert systems.

Progress is also slowed because Ph.D. candidates are expected to work independently; the design and building of any significant NL or expert system generally cannot be split very many ways while still allowing suitable academic credit to be assigned to all participants. Once a system has been shown to be feasible, there are further difficulties; universities are not in a good position to develop or support software, since the designer/builders generally leave after receiving a degree, and there are few academic rewards for cleaning up someone else's system and making it robust.

Industries have special problems as well. The largeness of NL and expert systems, and the fledgling state of our current technology makes it difficult or dangerous to promise profitable products on any time scale short enough to be appealing to management. There is also a wide gap between many of the kinds of programs that are produced in academia and the kinds of programs that can become good marketable products. For example, while story-understanding programs are considered (rightly) an important current academic topic, such programs have no clear short-term market potential.

The comments in this section so far have assumed that we will continue to do research in roughly similar ways to those that have been employed in the recent past, that is, that we will attempt to build AI programs that are "instant adults", systems which are the result of programming, not of learning. There has been a growing interest in learning over the last few years, although only a fraction of the learning research has been directed toward NL and expert systems. Certainly there are serious difficulties with the engineering of a system capable of learning NL or expert knowledge from experience (an "instant infant"). For NL, the most serious problem seems to be adequate perceptual apparatus for extracting "important" items from

raw sensory data. Even if we could devise such a learning program, there is little reason to suppose that it could be programmed to learn language much more rapidly than a human can, which would mean that at least several years would have to pass before we could judge whether or not the system was adequate. And, of course, the chance of getting everything right the first time is close to nil. Nonetheless, this seems to be an important avenue to explore, to hedge our bets, to further cognitive science, and perhaps to find a compromise position ("instant five-year-old?") that would represent the optimal long-term route to fully general NL systems. Further reasons for emphasizing learning include the observations that the only language users that we know are built through learning, and that we continue to use learning as adult language users, so that we probably need learning of some sort in our programs anyhow.

For expert systems, a key difficulty is that human experts are generally not very good at explaining the basis of their expertise. We are very far from knowing how to design programs that could build a suitable knowledge base and body of inference rules by simply watching problem situations and the expert's solutions for them. Again, such a system would have to have sophisticated ways of judging what items from its experience were important, and of inferring the nature of the knowledge of the expert from these items. This seems impossible without beginning with a highly structured system, about which we currently have very few concrete ideas.

Another practical problem is finding good application areas, that is, ones where NL or expert systems can be truly helpful, where the domain is well-circumscribed and well-understood, where the tools that we now have are capable of conquering the problem, and where typing is possible and acceptable as an input medium.

Finally, there is an acute shortage of qualified researchers. Since most actual application programs are likely to be the result of development by industry, it seems desirable to encourage more industrial effort. However, taking researchers away from universities reduces the number of faculty capable of supervising graduate students and carrying on much-needed basic research.

4.1 *Manpower*

AI shares a severe manpower shortage with the rest of computer science and computer engineering. However, in AI the problem is compounded because a "critical mass" of researchers seems to be essential to carry out first-rate research. To be concrete, let us list some of the requirements for doing NL or expert systems research.

Software support - Since AI uses languages and systems software packages and often machines that are different from the rest of computer science, separate systems people are extremely important. At least one, and preferably two or more are needed. Implementation is essential: ideas cannot be tested without implementation, and many ideas result from the experience of implementation. The day of the single-researcher tour-deforce system has largely passed. Because implementation is a lengthy and costly process, it is important to argue before implementing. In arguing, the generation of new paradigms is important -- a re-implantation of previous ideas is of little value.

Within NL, a modern system should have a user interface, a parser, a semantic interpreter, a knowledge representing and retrieving component, a discourse plan evaluator, a speaker modelling component, and a language generator. In addition, if it works on a data base or knowledge base of some kind, there is a need for further support. There are 6-7 different areas here, and even if each peson is expert in three, there is a need for four or five people. The story is similar for expert systems: an expert system must have a large knowledge base, which generally requires at least one expert and one knowledge engineer. An expert system also requires a user interface, which involves both input understanding and output generating components. Again the minimum critical mass is four or five researchers. Using senior grad students is possible, but turnover after graduation causes serious problems in keeping a system working. Using these criteria, only three or four U.S. universities have a critical mass in NL, and a similar number have critical mass in expert systems.

People who want to do AI research must also be cognizant of fields outside computer science/computer engineering: NL people ought to know linguistics and possibly psychology and philosophy of language; expert systems people must know about the knowledge area (medicine, geology, molecular biology) in which their systems are to be expert; vision people need to know our neuro-physiology and the psychology of perception; and so on. This means that effective critical masses may be even larger, and may only be possible in settings which can provide the right support outside of computer science/computer engineering.

Leaving the critical mass issue, there is a critical manpower shortage at universities, especially for research and for graduate research supervision. Few new Ph.D.s are being produced, and computer science suffers from a singularly high emigration to industry (see Kent Curtis' summary); part of the problem is teaching loads. In a time of rapidly expanding demand for education in computer science/computer engineering, both for specialists and for those who want computer science electives, teaching loads are likely to be high. This is compounded by heavy graduate advising demands. Of the six faculty members in our group, one supervises twelve graduate students, two supervise ten each, one supervises six, and one supervises two. However, the one who supervises two also supervises eighteen full-time professional staff members and the one who supervises six has a relatively heavy teaching load. When the need to write proposals for the substantial equipment and research funding needed to support serious research are added, it is clear that faculty in AI may well have difficulty fitting in time for research. Graduate students have in some cases decided not to go into academia specifically because of what they have observed of their advisor's experience.

Industry too has problems, many not specific to AI. There is such a shortage of qualified personnel that nearly as many non-CS majors as CS majors are hired for CS positions. CS education at universities also varies wildly in quality, especially at minor institutions, and there is no easy way (short of some form of profession certification) to judge the competence of prospective employees. No doubt, many of the problems in generating high-quality software stem from poor education.

4.2 *Equipment*

AI research requires extensive computer facilities, generally different types of computers than those used for large numerical tasks. There is a wide gap between the

top few universities and research labs and the rest of the universities doing AI research. The top several AI centers now have one powerful, dedicated computer (usually LISP machines) for every three or four researchers (graduate students and faculty) as well as a number of larger time-shared machines, ARPANET connections, and other special-purpose equipment.

If reasonable AI research is to be done elsewhere, much more equipment must be made available. More equipment is also critically important to keep new Ph.D. graduates in academia: a number of industrial labs are at least as well equipped as the best universities. Graduate students are unlikely to want to go to a place that has computer facilities that are inferior to those they are accustomed to.

AI researchers at universities which do not have software support groups are now constrained to use a relatively narrow range of equipment types in order to use shared software from other locations. The manufacturers of this preferred equipment (DEC, Xerox, Symbolics, LMI Inc., and a handful of others) have not donated much equipment in recent years; the few exceptions have been universities that are already relatively well-equipped. The future looks a little brighter, since a fair body of AI software can now run under UNIX, and many manufacturers either offer, or plan to soon offer, machines (especially based on the Motorola 68000 chip), that can run UNIX. Maintenance funds are also critical; there have been cases where industrial gifts have been turned down by universities because maintenance was not included, and the universities had inadequate budgets to pay the maintenance costs.

5.0 *AN ASSESSMENT OF THE RESOURCES REQUIRED FOR AI RESEARCH*

AI's main needs are manpower and equipment. Except for very limited special purpose applications, AI systems require relatively large program address space and relatively large numbers of machine cycles. When done on the time-shared machines in general use (e.g., Digital Equipment Corp. VAX's and DEC-20's), research is clearly hampered by a shortage of machine power. LISP machines are much better, but some (e.g., Xerox 1108's at $32K each) have rather small address space, and the rest tend to be quite expensive (more than $75K) if dedicated to a single user. Very roughly about $25K-$35K in additional funds per researcher is needed to provide at least a semblance of the best environment for AI research; anything else will mean that progress is slower than it could be. In this estimate, we have assumed that a number of researchers already have LISP machines, that some researchers will get more expensive machines (Symbolics 3600, LMI Nu machine, or Xerox Dorado), and that shared facilities, terminals, modems, printer, mass storage, etc., must be purchased also. Including graduate students and research programmers, as well as faculty researchers, there are now probably 150 people doing work on NL and expert systems in universities. This brings the total expenditure needed to properly equip these researchers for maximum progress to about $5M. About $50K-$60K is required to properly equip each new faculty member.

Manpower needs cannot be solved quickly by the simple infusion of dollars. Money that is spent on university equipment will probably do the most good, because it can help speed the research for graduate students, and it can also make universities relatively more attractive places to induce faculty members to remain and new graduates to choose in place of industry. This will in turn accelerate the production of new researchers. At the best, we are still likely to have a serious shortage of AI researchers for the foreseeable future: given that there are fewer than 25 faculty members who each graduate about 0.5 Ph.D. students per year, we can expect for the near future only 12 or so new NL Ph.D.s per year. In expert systems, the situation is a little better, with about 30 faculty nationwide, but there will probably still be no more than 15 new Ph.D.s per year over the next ten years. Probably only about half of the new Ph.D.s will go to universities, and it will be five years before the new faculty produce their first Ph.D. students. Thus the manpower situation for the next ten years is likely, in the absence of any massive intervention, to leave us with fewer than 100 NL faculty and about 100 expert systems faculty members nationwide at the end of the ten year-period.

As mentioned above, a possible way to increase our AI capability in the shorter term would be to encourage a cross-over of faculty from areas such as linguistics or development psychology to AI NL research, and from various fields of expertise (e.g., medicine, geology, etc.) into expert systems research. There are already incentives for researchers with suitable backgrounds, since funding in many other areas (e.g., linguistics) has generally been cut along with social sciences. An infusion of researchers from these areas may have possible long-term benefits to AI NL research above and beyond providing more manpower; it is our belief that in order to truly succeed in producing general, robust NL systems, we must develop a far deeper science of human language understanding and language development, and it seems clear that expert systems research requires humans possessing the specific expertise to be part of the effort. In addition to utilizing faculty members from other areas, it may be advantageous to gain the assistance of appropriate researchers in industrial laboratories in supervising doctoral research.

It might also be helpful to provide graduate fellowships, though the most serious shortage seems to be supervisors of research, not interested students nor money to support the students.

6.0 *RECOMMENDATIONS FOR THE NEXT 5-10 YEARS IN NATURAL LANGUAGE*

(0) Don't expect too much; the number of researchers in the field is really quite small, and the size of the task of understanding is enormous.

(1) Continue a broad range of basic research support; there is still a shortage of science on which to base NL system engineering.

(2) Increase funding for equipment. Adequate equipment is essential if researchers are to produce working systems, rather than just theoretical advances. Such funding has many benefits: Increasing available computer power can dramatically cut the amounts of time and effort required to produce running systems, since easy-to-write though computationally expensive systems can then be considered. Squeezing a large program onto a small machine can be very time-consuming. And if programs take too long to run, programmers start to work on speeding them up instead of working on increasing their range of competence. Making modern, powerful equipment available to universities will help them retain faculty. Despite the

apparently high initial cost compared to salaries, money spent on hardware is likely to be a good investment.

(3) Encourage resource-sharing by funding specific research groups to develop supply, and maintain a common body of research tools, for example AI programming languages, natural language parsers, knowledge representation systems, "empty" expert systems (i.e., reasoning and knowledge base access portions of expert systems with domain-specific knowledge removed), and programs for transforming programs from one language or operating system to another. To some degree, this recommendation is already being followed, and has accelerated research progress.

(4) Create and encourage development groups in industry and military labs, and encourage increased contact between such groups and university and industrial basic research laboratories. Universities are particularly ill-suited for developmental efforts, since there is a high turnover of key system builders, making it difficult to support application systems. In addition, we need all the effort on basic problems that can be mustered. Development could be handled by groups that have more traditional software backgrounds; once feasibility has been demonstrated, AI systems often look a lot like other programs. Possible incentives could include tax breaks and jointly funded university/industry research and development efforts, though the latter would have to be designed judiciously to avoid waste and mismatched expectations and capabilities.

(5) Encourage industrial research laboratories to help by advising Ph.D. research wherever possible. Such cooperation can benefit the laboratories by providing relatively low-cost, high-quality staffing, and can help increase the size of the U.S. research community at a faster rate. This recommendation can work; SRI International and Bolt Beranek and Newman, Inc., among others, have successfully functioned as Ph.D. research supervising institutions for a number of years.

(6) Maximize faculty research and research supervision time by providing partial academic year salaries as well as summer salaries.

(7) Institute three-to-four year research initiation funding, including equipment funds, for promising new graduates who agree to stay at universities. Hewlett-Packard has already undertaken such a program.

(8) Encourage the design of novel supercomputer architectures that take AI needs into account; current supercomputers are designed for numeric computation and are of little use or interest to AI, though AI badly needs computers with greater power. AI researchers have begun to design such machines at a few locations. AI people have had some success at computer design (e.g., LISP machines), but it would be desirable to have groups that specialize in computer design become involved with such designs. Japan, in its Fifth Generation Computer effort, has already undertaken such a goal, and it seems quite possible that even a partial success in their effort can cause serious erosion or loss of the U.S. high-tech edge.

(9) Encourage cooperation between AI NL research and the traditional fields interested in language (linguistics and psychology). This can serve the purpose of aiding the building of a science of language and cognition and can also provide a more rapid increase in manpower resources than is possible through the training of new researchers only. This recommendation may happen anyhow, because "social science" funding has been cut dramatically, putting pressure on many of the people that could help the AI NL effort. To make this work, a fair amount of re-education, especially in AI, computation and programming, will be needed along with some re-education of AI people in linguistics and psychology, to build a common knowledge base.

DISTRIBUTED COMPUTING AND NETWORKING

An Assessment of the State-of-the-Art and
Recommendations for Future Directions

Prepared for
NSF Information Technology Workshop
XEROX International Center
Leesburg, Virginia
January 5-7, 1983

by

Harold S. Stone, Chairman	University of Massachusetts
David J. Farber	University of Delaware
Robert Gallagher	Massachusetts Institute of Technology
Oscar N. Garcia	University of South Florida
Hector Garcia-Molina	Princeton University
Lawrence H. Landweber	University of Wisconsin
Edward D. Lazowska	University of Washington
Rick Rashid	Carnegie-Mellon University
Robert Thomas	Bolt, Beranek & Newman, Inc.
Andries van Dam	Brown University

DISTRIBUTED COMPUTING AND NETWORKING

TABLE OF CONTENTS

1.0 *INTRODUCTION*

There is overwhelming agreement that because of continuing hardware advances, personal computers and professional workstations will become as ubiquitous as home color television and office telephones by the end of the current decade. Already sales of small personal computers are nearly a million a year! On a university campus or in an industrial or governmental organization, this explosion of computing will mean that there may be literally thousands of machines within the organization, essentially one for each "knowledge worker" and support person.

Extrapolating current trends, these machines will not be stand-alone stations but will be electronically connected to a hierarchy of local area, backbone, and geographically dispersed long-haul networks. The key question is not how to connect machines electronically, but rather how to do it at higher levels, for purposes of sharing both information and processing. Indeed, the question may be phrased as "what does it mean to have the ability to connect millions of machines together" - how will individuals work (and play) together, how will work and information be distributed, shared and protected? What new industries, service providers, technical specialties emerge? While answers to these questions are being developed, both in the local-area network and longhaul network research communities, it is clear that the complexity of the problem grows enormously as we move from networks with several hundred machines to those with millions. It is likely that entirely new ways of thinking about systems of this size must be developed, much as modern electronic telephone switching bears little resemblance to switchboard-panel technology.

Today's growing proliferation of machines, networks, and user environments constitutes a "tower of Babel" threat to mar the rosy picture of the "wired" campus, city, state or nation. Unless the difficult research topics listed below are tackled with vigor to yield pragmatic answers in the next 5 - 10 years, and unless there is greatly increased interest in and compliance with standardization efforts, there will be chaos; users will not be able to communicate effectively. Research in such questions as resource management, user environments and privacy/security must continue for classical centralized systems, for small (less than 100 processor) systems, for tightly coupled distributed computer architectures, and for (inter)networking; these same issues need to be tackled as well in the vastly larger context of the "megacomputer network" of the future.

While industry will clearly be involved in research and development in this exploding area, basic research is vital to provide answers to fundamental questions and to provide directions for development. With industry/university partnerships developing in the research area, there is greater likelihood of telescoping the time needed to develop successful distributed systems.

We see gaps in present knowledge, which if filled, could make enormous differences in the effectiveness of mega computer networks. Among the critical issues are the following:

1.1 *Innovative Uses*

Computer networks of one million nodes or more present opportunities for innovations of an unprecedented nature. When networks were first implemented at the end of the 1960's, their immense value to support electronic mail was completely unforeseen. This in turn has given way to teleconferencing, whose conception is due completely to the emergence of network technology. To draw an analogy from the VLSI community, consider the development of the microprocessor, which might have been foreseen prior to its development. Then consider all of the resulting fallout of the microprocessor which has made economically feasible the technology of office automation and the home computer. This secondary effect of VLSI technology was unimaginable in the years prior to the development of VLSI.

Clearly, the million-node network creates opportunities to gather, collate, and disseminate information in totally new and innovative ways. Research funding for innovative applications well in advance of the implementation of such networks would greatly influence their implementation and assure that their eventual development could support

imaginative and important new ideas.

1.2 *Problems of Scale*

We are confronted with the prospect that within a decade the size of our systems will grow by several orders of magnitude. This explosive growth renders obsolete much of what we now know about designing and controlling systems: - It is not possible to observe the global "state" of such a system using a conventional definition due to uncertain time delays and the likelihood of error in message-based communication. - The algorithms that we presently employ are infeasible for systems of the size with which we will be confronted, due to greater than linear growth in processing and data storage requirements. - Our intellectual ability to "visualize" the system fails, due to it immense size and asynchronism. For the same reasons, existing aid to comprehension also fail.

An analogy to the problems currently being experienced in the area of design aids for VLSI circuits may be helpful in understanding the problems of scale. In a very short period of time, the number of gates per device went from 10,000 to 100,000, and devices with 1,000,000 gates will be achieved in the near future. This revolution has created enormous opportunities as well as enormous difficulties. The approaches to designing and verifying devices with 10,000 gates were stretched to their limits with 100,000 gates, and are simply inadequate to deal with devices of 1,000,000 gates. This inadequacy is felt in many ways. New techniques are required to make designs of this size modular and fault-tolerant. Design-automation algorithms fail on 1,000,000 gate designs (for example, it can take many days on the fastest available processors to do logical and electrical verification of a design.) The intellectual ability of designers to grasp designs of this size fails, and existing design aids provide totally inadequate assistance.

The problems now being confronted in the area of VLSI design are very similar to the problems about to be faced in distributed system design. The key intellectual issue is how to "decompose" into natural clusters an extremely large, extremely complex system so that its complexity can be managed -- so that we can grasp its functional units and their states, so that algorithms can be designed to control them, and so that the system as a whole can be visualized and comprehended. In addition, we must devise robust algorithms for controlling these systems -- algorithms that "converge" in the presence of imprecise information.

1.3 *Destandardization*

The present trend towards a great diversity of computers, software, communications interfaces, and communications protocols has led to some difficulty in interconnecting a variety of systems together into one computer network or into a closely-coupled distributed processing system. Where ad hoc solutions have worked effectively today, the problems of the future will be greatly exacerbated by the trend toward greater diversity. Standardization can alleviate some of the problems, but is not a satisfactory total solution when advances in technology make existing standards obsolete within a few years of their introduction. What is needed is a better understanding of how to cope with diversity, and this in turn requires a better understanding of the nature of networks and distributed systems. To solve the problem, it is essential to know the objects and processes that are the basic building blocks of individual nodes, and then to determine how to transform those of any one node into those of any other node. This holds as well for the communication protocols between nodes, which undoubtedly will differ from across a very large network. Not only must conversions be made automatically between differing segments of a network, but it must be possible to define and attack a totally new type of node as well.

2.0 *THE RESEARCH AGENDA*

There are several fundamental problems of distributed systems that must be attacked to create a body of knowledge to support the next generation of distributed systems. There have been notable advances in the past that have led to successful commercial exploitation of distributed systems. But to a large extent, the speedy exploitation of available technology has leap-frogged over the basic questions. Fundamental problems must be addressed to support future technological advances and to explore more deeply the promising, but difficult, areas that have been bypassed. Good ideas across the whole spectrum of distributed computing merit attention. The following problem areas describe some of the major areas of research questions today. The list is not intended to be complete, nor is the order indicative of relative priority.

2.1 *Theoretical Foundations*

Distributed processing lacks the theoretical underpinnings that exist for traditional sequential processing. Because sequential processing has an acceptable model of computation, one can pose and answer questions of the limits of computability and the complexity of computation. Distributed computation lacks an adequate model to date so that it is not even clear how to pose such questions. The essential difference between sequential and distributed processing is that distributed processing has uncertainty associated with the value of remote information and the state of remote devices. A processor can read remote devices to obtain information on data or state, but by the time that information reaches a processor it may be obsolete. How does one model this property? Given an adequate model, how does one pose and solve the interesting questions concerning control and data access that make this model so different from conventional computation? Complexity measures and theoretical and practical limits on the computational power of a distributed system should be directly available from a suitable model of computation.

A good theoretical model also should be helpful for verification of distributed programs, but the verification process for distributed programs is itself an important open problem. From a practical point of view, distributed-program verification may have to be supplemented by testing procedures. Here again, theoretical research may lead to testing and verification procedures, which in combination yield satisfactory means for validating distributed programs. It is necessary to balance the cost of exhaustive verification against nonexhaustive testing. The following considerations suggest why a combination of verification and testing may be effective.

1. Verification can be done without building and running a distributed program, which could be a costly process. Consequently, verification can give confidence in the correctness of a distributed process before committing resources to implement it.

2 For practical reasons, testing must often be nonexhaustive. Hence, an algorithm that passes all tests may still have flaws. Verification may reveal what flaws are detected by specific sets of tests.

3 If an algorithm satisfies a nonexhaustive battery of tests, verification may reveal the limits on the corresponding domain of inputs for which the tests guarantee correct behavior. Therefore, theoretical studies into both program verification and program testing techniques could yield results of great utility. We note that testing and verification techniques of distributed programs appear to be more difficult than conventional programs because of inherent characteristics of distributed programs with regard to access to remote data and state information.

2.2 *Reliability in Distributed Systems*

Reliability is one of the factors that has pushed the development of distributed systems. The idea is that an entire centralized system fails when the central processor in that system fails. Presumably, the failure of a single node in a distributed system does not harm the operation of the remaining nodes. But design techniques to achieve this goal have not been perfected, although "nonstop" computers show that techniques have reached the stage where they can be exploited commercially. Note that the nonstop computers are relatively simple systems as compared to the huge network of computers that are likely to exist in the future. Reliability of a complex network may be more difficult to achieve because of the problem of scaling.

Other approaches toward reliability may use a mix of techniques. What techniques, for example, are most suitable to achieve ultra-reliability? Can the cost of ultra-reliability be greatly reduced with only a small decrease in reliability? Can a specific portion of a system be given ultra-reliable protection and have its protection unimpaired by other less reliable portions of the network? If so, which portions of a network should be made ultra-reliable to achieve a reasonable trade-off between total cost and total system reliability?

The possibility of obtaining incremental changes in reliability of a distributed system for small incremental cost increases is an important factor here. This may be a very attractive cost-effective method for gaining reliability. But much depends on research into the reliability techniques and the identification of where they should be applied. Large distributed systems might well exhibit a failure mode that is a form of system instability somewhat different from failures directly attributed to component failures. Instability may arise from deadlock, from saturated communication links, or from software faults. A reliable distributed system must be free of instabilities, as well as resistant to hardware failures. Therefore, in the presence of an error or fault condition, a distributed system must be robust and resilient, so that the perturbation is contained and cannot propagate through the system. Moreover, it must be possible to isolate faulty nodes from a distributed system for repair without interfering with the computation taking place in the remainder of the system. For otherwise, the repair of any node may force a total system shutdown, which becomes intolerable as system size increases. In a large system, successive node removal may suddenly disconnect some part of a system from another part. How can this be prevented? If not, what are the ramifications and how does recovery occur?

A major goal for research in the design of reliable distributed systems is to be able to assure that when a message, file or a transaction is committed to the system, the message or file will arrive at its destination without error, and in a timely fashion. The user should be able to assume that, unless informed otherwise, the system has carried out the request correctly. If the system is unable to complete a request, the system must ensure that the transaction is fulfilled in its entirety or not at all, because partial fulfillment may leave the system in an incorrect state. The techniques that have been classically available in multiprocessor systems for this purpose need to be extended and improved to manage the recoverability of data and processes within a distributed system. Not only should one expect a node in a distributed system to be capable of recovering from its own malfunction, but also nodes should be capable of detecting the malfunction of other nodes to take corrective action to prevent further damage. A gracefully degraded system under these circumstances should continue to be operational.

Distributed computers are unlike centralized computers in regard to recovery because of the observability of state. A centralized computer can be restarted by setting its initial state, and it can be checkpointed by saving its state at some intermediate point. It may be impractical to set an initial state in a distributed system, especially as the number of nodes grows as large as one million. Moreover, it may truly be impossible to save its state. Hence, we must find ways to recover. For example, we might save other "snapshots" of consistent states of fragments of data bases, hoping to install them later when necessary. There are many open questions here, including the applicability of the technique to networks with a million or more nodes. In any event, the solutions in place for multiprocessors are not directly applicable to distributed systems.

2.3 *Privacy and Security*

In distributed systems, as in conventional centralized systems, there is a need to protect a user's stored data, and to provide means by which a user can exercise control over access to protected data. In addition, in a distributed system, there is a need to protect data in transit from one component to another. Hence, there must exist a mechanism for naming a user, for distinguishing one user from another, and for authenticating that a user who claims to be User A is, in fact, User A. These requirements hold for centralized systems as well as for distributed systems. But in distributed systems, there is the additional difficulty in dealing with remote access to cryptographic keys, protected data, and authentication processes. Can this really be made secure? Some of the questions of interest and worthy of investigation are the following:

1 The existence of user ID's implies a user registry of some sort, whether actual or only conceptual. How should such a registry be organized? Should it be centralized? Distributed? Replicated? How should entries be added to and removed from the registry? Who should be permitted to make such changes to the registry?

2 What sort of authentication procedure or procedures should be used? Passwords? Voice prints? Challenge/response? Card keys? If a password

mechanism is used, how can passwords be protected? What system entities can be trusted to perform authentication? At what point should authentication be performed?

3 What sort of access control mechanism should be used? How can access control be implemented in a reliable fashion? The effectiveness of access control mechanisms depends upon the extent to which they are actually used. What sort of user interfaces to the access control mechanism should be provided?

4 The existence of mechanisms to ensure the privacy of data against inspection by the unauthorized user may invade the user's privacy because they can be used to monitor the patterns of the user's activity. How should the system be organized to prevent unauthorized monitoring of a user's activity?

5 What grades of service for protection can be offered?

6 How do you offer access to aggregate data while protecting individual data?

2.4 *Design Tools and Methodologies*

The specification, design, and implementation of a computer system is a difficult exercise in mastering complexity. Tools that assist in the tasks of specification, evaluation, and debugging are essential to this process. The need for development methodologies is especially important because our intuitions fail us in the case of large concurrent systems. Every system must meet goals of functionality, performance, correctness, and reliability. Whether these goals are ambitious or easily achieved, it must be possible to design them into a system, rather than add them as an afterthought. One key requirement is the ability to predict performance prior to implementation. Another is the ability to validate or verify a program from its specification or its behavior on test cases. Initial progress has been made in both of these areas, but the scale of the systems that we will be called upon to construct in the foreseeable future demands considerable advances.

In the same way that users require high-level linguistic support, so do system designers and implementers. Programming languages must be tailored specifically to the distributed environment by providing for interprocess communication and the control of concurrent operations on shared data.

Facilities for testing and debugging clearly are important aspects of building real systems. Testing and debugging typically are accomplished by controlled execution of a program in a test environment and involve starting, stopping, inspecting, and continuing the program. These functions are extremely difficult in a distributed environment for a number of reasons, some of them practical and some of them fundamental. In the former vein, there are more computing elements to control, and it is difficult to start and stop a distributed program in a coordinated fashion. In the latter vein, it is a challenging problem even to define what we mean by the "state" of a distributed computation, much less to capture that state and restore it during the debugging process. The ability to monitor and control communication between cooperating components is a useful mechanism for testing and debugging, but more powerful and flexible techniques need to be developed and integrated into debugging tools.

Finally, the majority of the evaluation of mechanisms for controlling distributed data and distributed programs can only be conducted experimentally today. On the one hand, modeling tools must be developed to reduce the need for this relatively costly approach. On the other hand, we must accept the fact that experimentation is a valuable exercise, and develop efficient approaches for carrying out experiments. One approach, the "testbed," emphasizes modularity and flexibility, so different algorithms and strategies can be easily "plugged in" and compared.

2.5 *User Environment*

There is a rapidly growing familiarity with educational and video games, office automation tools, such as electronic mail, personal databases, spread sheets, and word processing, and even teletex/videotex information services. After many years of rapid development, the utilities available for distributed systems at the end of the decade will be far richer, far more diverse, far more comprehensive than the current state-of-the-art for stand-alone systems. Again, the problem of managing complexity arises, this time at the user/machine interface, the port to the global system. Most users will not be trained in computing, and the interface must, therefore, be unlike today's predominantly programmer-oriented computer interfaces. As those constructing office automation systems have found, it is critical to design services "outside in," that is, starting with a common user interface, rather than with individual subsystems integrated *ex post facto*. Research is required into how such interfaces should be specified, validated, implemented, documented, taught, and grafted onto existing programs that *must* be preserved and cannot be rewritten.

It will be a design goal of distributed operating systems to present to the application programmer the abstraction of a "virtual uniprocessor" supporting multi-tasking for many purposes so as to hide a complexity of distributed resources. This involves, among many facilities, the acceptance of a common representation of data structures and their operations and/or automatic conversion services. Conversely, for other applications, especially for those in which performance is paramount, the multi-window user interface should make it relatively straightforward to specify the linking of resources and to monitor the progress of the distributed computations, each reflecting its status in its own window. A graphics-based, multi-window environment will allow the user to participate in a multiperson interaction, read incoming mail, reply to prior mail, etc., all without switching modes.

There will be two types of programming: where possible, users should be able to combine existing utilities with a powerful command interpreter, with a primarily pictographic ("iconic") interface or specify new programs in a high-order "visual programming language." This language will likely emphasize programming by example ("let me specify how to do it"). For example, message-based interprocess communication might be specified by indicating icons representing the processes, having mailbox icons be attached to their corners and picking the variables to be transmitted from a menu associated with a process icon.

While tree-structured directories and menus are well-known, they are not sufficient to handle the number of options and sizes of databases that we may expect. New

selective dissemination of information techniques will have to be invented, in conjuction with expert systems which can dynamically tailor the presentation to the user profile. Also, graphical specification of executive commands and programming language constructs is another young research area requiring not only the development of new user interface techniques, but also research into underlying formalisms that allow "behind the scene" verification and optimization of user command sequences.

2.6 *Distribution and Sharing*

In theory, distribution of data and programs can bring about higher performance and reliability, but in practice, achieving these goals for a specific application is a difficult and time-consuming task. For example, a program can be partitioned into a number of concurrently executing modules to improve its execution time. This has been done successfully on some applications, such as sorting, but to exploit fully the potential advantages of distribution, we need to develop *general* mechanisms for decomposing problems, replicating resources, and distributing work load across a system. Strategies for managing data and programs in a distributed system must be devised. For instance, program segments must be synchronized and scheduled. They may have to share common data, and the consistency of such data must be preserved.

Currently, the decomposition, replication, and distribution operations have not been automated, so that system implementers have a difficult task in preparing programs for distributed systems. Decomposition appears to be particularly difficult because it depends strongly on program behavior factors that are made evident during program execution and are difficult to estimate from a program specification alone. The state-of-the-art today for decomposition, replication, and distribution taken together is largely a matter of experimentation on trial solutions followed by the selection of the best one. This methodology clearly cannot be used for the million-node network.

There are actually two different motivations for distribution and replication of modules across a distributed system. One motivation is enhanced performance; the other is reliability. Automatic algorithms for decomposition, replication, and distribution are undoubtedly quite different when driven by reliability considerations, as compared to performance considerations. The open question here is to invent algorithms driven by each consideration individually and then by both considerations jointly. Just how they are affected by treating both reliability and performance jointly, is still a matter of conjecture.

Other important problems that arise in the control of distributed processes are:

1 Scheduling of tasks;

2 Synchronization of the start of one task to particular events, such as the completion of some other task;

3 Deadlock;

4 Temporary inconsistency of data.

Many of these problems first manifested themselves in centralized systems. Often, the mechanisms developed in that context form the basis for solutions in a distributed environment. There are significant complicating factors, however, such as the lack of centralized control information, the presence of physical concurrency, the potential for partial failure, and the replication of data and processes. Despite these difficulties, mechanisms for distributed systems have been found recently for a number of these problems. Future research effort must be devoted to developing techniques for evaluating alternative mechanisms, and to developing new mechanisms that can be scaled to accommodate very large systems.

2.7 *Accessing Resources and Services*

The ease with which a user can discover the existence of resources and services, locate them in the internetwork, and access them will determine the nature of the cooperation between users in a large distributed system. The ARPANET community, for example, has consistently been limited in its ability to share resources and services because the host operating systems of the machines on the network do not provide the necessary tools either to the programmer or to the user to build distributed applications. Interaction on the ARPANET has, therefore, consisted almost entirely of mail, file transfer, and remote-terminal access.

The need for closer interaction within the ARPANET has not been a pressing one, since each machine is usually an autonomous timesharing system complete with secondary storage, printers and a large user community and maintenance staff. As our computer networks grow from hundreds to hundreds of thousands of machines, each individual computer supports a smaller user community (eventually a single person or family) and becomes much more specialized - no longer providing locally services, such as large secondary storage, archival storage, printing, data acquisition, and database management.

This specialization makes the need for simple access to distributed services a critical component of the success of the network. Unfortunately, the potentially large number of computers involved also drastically increases the complexity of providing for simple access. Directories of services, for example, potentially become enormous. When equivalent services are being offered in a number of different locations, algorithms must be developed to select the appropriate instances of a service. When many such services are required to perform a single task, it is necessary to orchestrate the resource allocation.

The resources and services available in a distributed system are potentially represented and treated differently, yet functionally equivalently, throughout a distributed system. Thus, when a user invokes a service, the effort will be the same, regardless of which processor performs the service. This is a major challenge in a heterogeneous large-scale system. Let us consider some of the problems that arise in such cases.

1 *Basic Notions of Resources and Services*

What resources and services should be provided? Some are obvious and include file access, archiving, and database access. Some are less obvious, and might include the ability to treat, for example, a screen image file as an entity that could be moved, displayed, or transformed.

What are the most elemental services and resources? How can they be represented throughout a large network of dissimilar

computers? Is it possible to achieve machine-independent access to these resources?

2 *Discovery*

Mechanisms need to be provided for determining what services are available on the network and to determine their specifications and interfaces to other programs. In essence, this is the problem of providing on-line "yellow pages." Mechanisms such as the UNIX on-line manual facility already exist for single computer systems, and attempts have been made to provide on-line manuals of services for the ARPANET community. A number of potential problems arise when considering extending such services to a very large distributed system.

3 *Naming and Locating Services*

Once a user's program requires a specific service, there is the problem of resolving a name for that service into a location in the system and an access right to communicate with that service. Most name-lookup mechanisms are local in nature and are of limited complexity, but some are global and have to deal with problems such as parts of the name space under different administrative controls, indefinite expansion of the name space, and extremely large name spaces.

4 *Accessing Services/Interprocess Communication*

At the programmer's level, a uniform interprocess communications mechanism needs to be provided, and must be powerful enough to handle both synchronous and asynchronous styles of interprocess interaction. Ideally, the interprocess communication facility can hide the diversity of networks that will inevitably coexist. It is equally important to be able to transmit and receive access rights to invoke services as well as to invoke the services. Current interprocess mechanisms provide these facilities for 100 or fewer nodes. The solutions used in these implementations are unlikely to scale to serve very large networks.

2.8 *Distributed Databases*

A distributed database system allows users to access and to share a community of databases, and will form a critical component of many megacomputer networks of the future.

Distributed database systems are in themselves distributed systems, so all of the research questions covered so far appear in this research area, such as problems related to naming facilities, security, and reliability. In fact, several concepts that have been developed from within the distributed database framework are finding applicability in distributed systems in general. For example, the notion of control of concurrent operations on shared data first emerged in the database context, but was useful in proving the correctness of general distributed algorithms. Similarly, the executive of some distributed systems has been implemented using the idea of *atomic transactions*, which also originated in database systems.

In spite of this, there are a number of research areas that are specific to distributed databases, or at least do not appear in this form in other types of distributed systems. Here, we briefly outline some of them:

1 *Data Models*

How is the distributed data represented? How, if at all, is the user made aware of the data distribution?

2 *Heterogeneous Databases*

If heterogeneous databases exist, how are data commands translated? What model is used to view data in different databases? How are updates (if any) performed?

3 *Distributed Query Processing*

What is the best way to process a query that involves data at multiple sites?

4 *Multimedia Databases*

How can databases that contain text, graphs, pictures, voice, as well as conventional data, be accessed and managed in a distributed environment?

2.9 *Network Research*

There are several key areas that require attention. One of these deals with the architecture of advanced local networking systems that would allow integrated voice, data, and graphics transmission over wire, optical, and radio systems. Total system designs must provide for high availability and fault tolerance of all components, including the underlying communications system. Those two requirements are not achieved in today's local networks. In addition, the problems of extending the privacy-oriented protection systems which could exist in the processing elements of the distributed system into and through the communications system also is not understood.

Another open issue is the interconnection of networks with dramatically different capabilities, speeds, and protocols. Many examples of this will exist in the future and, currently, we have inadequate understanding of gateway design, protocol conversion, and capability-matching mechanisms. The potential of cellular mobile telephone to provide low-cost portable data communications services also raises many open and difficult issues in the addressing and protocol areas.

Protocol research has dealt largely with networks of independent processors which communicate text over moderate bandwidth lines. Very little work has been done on protocols which deal with more general objects and services, such as graphics images and integrated voice/data. Effective utilization of very high bandwidth lines also requires attention. It is not even clear that current transport and internet protocols will support adequately such enhanced services and communications capabilities.

Communication and resource sharing will be an important feature of the proposed environment. Design of processors and operating systems which facilitate protocol design and implementation is a largely untouched area.

Tools are required to aid in the design and evaluation of protocols. At present, design is largely an *ad hoc* process and performance metrics/tools are almost totally lacking.

3.0 *RELATED ISSUES*

Quite apart from the problems that must be solved to understand and build effective distributed-processor systems, there are problems related to their applications and management. The societal impact will clearly be extensive, since such systems can provide new capabilities, new modes of employment, and whole new industries. There are fundamental concerns, however, regarding privacy, security, and basic protection of the individual. Furthermore, the issues of protecting copyright of electronic publications, and of preventing theft of proprietary software and data, are arising. Should software and data be distributed in encrypted form, with the key provided upon verification of payment? Or should it be given physical form to make it harder to pirate, for example, distributing not just a read-only memory chip for the program, but a complete module containing a full microcomputer? How will language translation be handled for international cooperation? All these concerns have not yet been adequately resolved in existing systems and are a great deal more complicated in the distributed environment. It is necessary to obtain satisfactory answers to these questions in order to assure that the fruits of research and technology be used to the maximum benefit of a free society.

More pragmatic issues are related to the management of large distributed systems. They may become "information utilities," and within this context, it will be necessary to deal with the coordination of accounting, technical, and operational functions.

How will local systems be installed, diagnosed, repaired, and managed by the home, school or office staff? How will national databases be created and managed? How will evolution of the system be managed? Partial answers may be obtained from the videotex information and network services providers, but the potential numbers of machines, users, programs, and databases again compounds the problems dramatically.

Since these problems are different in kind from the scientific and technical problems discussed above, we believe that this new set of problems will be attacked by a different community.

4.0 *SUMMARY AND RECOMMENDATIONS*

This report has pointed out that fast-moving technology will result in the creation of multi-million-node computer networks by the end of the decade. This growth of several orders of magnitude will create enormous opportunities for innovative use. It also will render obsolete much of what we now know about designing and controlling systems. Research into fundamental questions of distributed processing is required to support the technology that will be in place in order to achieve its potential.

Among the areas of research of major importance are:

1 Theoretical foundations
2 Reliability
3 Privacy and security
4 Design tools and methodology
5 Distribution and sharing
6 Accessing resources and services
7 User environment
8 Distributed databases
9 Network research.

There are serious concerns about the resources available to conduct research in the field. Among the problems perceived are:

1 Level of funding
2 Manpower and education
3 Facilities.

While the first two concerns are important, there are no special requirements which distinguish needs in the distributing-computing-networking area from those of other computer science sub-specialties. Discussion of these relevant and difficult problems should, therefore, be held in the broader context of the field as a whole.

The question of proper experimental facilities for distributed processing is equally critical to the successful implementation of large systems. It has been amply demonstrated that hardware, software, research, and development are genuine experimental sciences, and that design ideas must be experimentally validated. A paper design, at best, predicts behavior/performance that may not reflect the characteristics of an implementation and, at worst, often cannot be implemented as is. Conversely, as in the physical sciences, experimentation often reveals unexpected phenomena and leads to new investigations. Given our minimal experience with small distributed systems with fewer than 50 processors and our total lack of experience with systems with more than 1000 processors, it is clear that it is vital to have experimental testbeds on which algorithms can be validated and tuned.

Such experimentation is an especially fruitful area of potential industry/ university collaboration, as the size and complexity of the facilities will put them beyond the means of a single organization. Both pre-existing smaller systems can be coupled and integrated, and brand new systems can be designed and constructed from scratch. It may well not be feasible to build an experimental net with 100,000 nodes, let alone one with a million nodes, so that special care must be taken to provide a convincing argument that the experimental results can be scaled up.

HIGHLY PARALLEL COMPUTING

An As[illegible]sment of the State-of-the-Art
Recommendations for Future Directions

Prepared for
NSF Information Technology Workshop
Xerox International Center
Leesburg, Virginia

January 5-7, 1983

by

James C. Browne, Chairman	University of Texas at Austin
Bruce Arden	Princeton University
Arvind	Massachusetts Institute of Technology
Forest Baskett	Digital Equipment Corporation
Bill Buzbee	Los Alamos National Laboratories
Mark Franklin	Washington University
Robert M. Keller	University of Utah
H.T. Kung	Carnegie-Mellon University
Duncan Lawrie	University of Illinois, Urbana
Fred Ri	IBM, T.J. Watson Research Center
Herbert Schorr	IBM, T.J. Watson Research Center
Howard J. Siegel	Purdue University
Lawrence Snyder	Purdue University
Robert Voigt	NASA, Langley Research Center

HIGHLY PARALLEL COMPUTING

TABLE OF CONTENTS

SUMMARY

Parallel computing is important because the power of uniprocessors is limited by basic physical constraints and because this limitation precludes effective application of computing to problems of great economic importance and of national security concern. The return of any *one* of the applications, e.g., truly accurate weather forecasting, dwarfs the total cost of development of very high performance parallel systems. Computing power is a fundamental pacing factor in the rate of knowledge growth in several fields of great intellectual and economic significance. Some of the fields, for example catalysis of industrial chemical processes, have become significant only when computers reached a certain stage of development. It can be expected that the increase in power obtainable from parallel computers will also open other new applications. The position papers of other panels from this workshop describe some of these applications and knowledge areas.

Parallel computing is a *systems* issue. The state-of-the-art for parallel software and parallel formulation of significant problems lags studies of parallel architecture. The issues of memory and I/O systems for parallel architectures lag the concepts of parallel computing engines.

This Panel report identifies and specifies the major problem areas for research attention in parallel computing and strongly advises stressing a truly scientific approach combining experiments and theory.

There is an abundance of concepts for parallel computing and abundant opportunities for developing these concepts into applications. The current bottleneck to progress is the difficulty of executing significant experimental studies. These experimental studies are essential to evaluation of total system concepts. The Panel has three recommendations for overcoming these bottlenecks. They are given in priority order following.

1 Establish facilities for the development and evaluation of feasibility prototypes of systems where modeling has established design merit.

2 Establish facilities based on existing parallel processor systems for development and evaluation of algorithms and problem formulations.

3 Establish computer facilities to support simulation (or emulation) evaluation of detailed models of proposed architectures.

The recurring funding for these facilities and projects should, after a buildup period, reach $50,000,000/year.

The Panel recognizes the difficulty of implementing our top priority recommendation of establishing Feasibility Prototyping Facilities and recommends that a workshop on this topic be held under the sponsorship of the NSF University/ Industry Collaboration Program.

1.0 *MOTIVATION, IMPACT AND APPROACH*

Interest in parallel computing is a natural consequence of the fact that the limit to uniprocessor performance is in sight. Improvements in circuit technology have left us with a set of diminishing returns in systems more than half of whose cycle time is consumed in signal propagation delays between active devices. Exotic technologies do not appear to have the capability of providing more than another order of magnitude improvement, if that. Highly pipelined machine organizations and vector operations deliver performance improvements whose degree and ease of attainability are both enormously variable.

It is apparent that uniprocessors will not be able to develop enough computational power to meet the performance requirements for major areas of application, including numerical simulations of complex physical systems, realtime signal processing, artificial intelligence and large-system emulation. The principal bottleneck is the limitation of a single processor to around one billion operations per second. It is further the case that this upper limit is theoretical and that technology factors may lower this limit by close to an order of magnitude. Parallel computation will be important not only because it can make a qualitative difference in the results obtained in the above application areas, but also because it will make possible applications in realtime control, applied artificial intelligence and analysis of complex systems. These new applications are ones where the economic leverage for productivity gain and technology development dwarf all previous

results from application of computers. The potential national security implications are equally as momentous. Parallel computing has the potential to revolutionize the computing scene because effective parallel computing may ultimately require fundamental changes in the ways we think about applications, languages in which to write applications and hardware structures in which to implement languages.

Effective parallel processing will require that there be achieved fundamental understanding of computational models, methodology, algorithms, language, architecture, processor-memory-I/O relationships, software, and interfaces to existing computing systems and end users. A systematic approach to highly parallel computation must embrace the following considerations. Exploration of fundamental issues and the understanding of applicability to end use are intertwined. Hence, the interplay of computation models, algorithms and architecture issues are important. System interfacing problems will be severe, as memory and I/O latencies and bandwidths cannot be expected to scale as rapidly as computational power. Software issues will be particularly important, not only because they always are, but especially because they are today less well understood than architectures. Extensive emulation and the reduction to practice of functional prototypes is vital to evaluate ideas, to provide qualitative and quantitative characterizations, to facilitate the study of dynamic behavior of application programs and to lay foundations for feasibility arguments. Since the constuction of high-performance machines is difficult and capital intensive, a facility which will be useful in evaluating several different architectures/languages will have a high payoff in the long run. An experimental facility will also provide flexibility which is not possible if a given architecture idea is directly committed to hardware. In many cases scalability arguments will have to be applied. In concert with either simulations or partial prototypes to adduce practicality, and the issues involved will have to be carefully thought out in advance. It is important in the short term to gain as much experience as possible on adequately supported parallel computing systems as they become available. Much of this work will be easier to motivate and evaluate if it is example-driven.

Citations and references have been purposely omitted from this report. There are a number of recent sources of survey material on parallel computers including special issues of Computer Magazine and the IEEE Transactions on Computers. The recently filed report of the panel (the "Lax" Committee) on Large-Scale Computing in Science and Engineering has a plethora of material describing needs for and uses of very high performance computer systems.

2.0 *RESEARCH PROBLEMS AND DIRECTIONS*

This section summarizes the research problems and research directions which the Panel perceives as being the most exciting and interesting in parallel computing and, at the same time, defines the core of knowledge which is required to bring development of *systems* 100 times faster than today's supercomputers. The underlining of *systems* in the previous sentence is to underscore that the issue is not one merely of parallel architectures but of usable systems with very high performance.

2.1 *Parallel Models and Algorithms*

There is no truly general-purpose parallel architecture to which all algorithms can be effectively mapped. This is unlike serial computation where almost arbitrary algorithmic structures can be mapped to give full utilization of a single processor. The challenge is to design parallel architectures that will effectively execute as wide a range of concurrent algorithms as possible. The second challenge is to design architectures which will very efficiently execute at least some significant algorithms. The dual facet of the second challenge is to design algorithms which will execute efficiently on a given architecture.

The essential foundation of knowledge can be determined through modeling the execution behavior of algorithms in the many possible models of parallel computation and their architectural realizations.

There are two fundamental aspects of this process. One is to define a taxonomy of parallel architectures and the basic structure of parallel models of computation. The second problem is to develop algorithms for significant computational processes, both numeric and non-numeric which effectively exploit parallel architectures. The synthesis of these two knowledge bases will provide the needed insights for effective design and implementation of parallel computations.

Models of parallel computations and parallel architectures must parametrically define at least the following factors:

a. the units of parallel execution
b. the mode of synchronization of unit operations or sequences of unit operations
c. the mode of communication: shared address space or message-based or both
d. the geometry of communication
e. the time of *binding* to hardware of a, b, c and d in a given architecture

The current understanding of all of these factors is incomplete. This lack of a complete foundation hinders mapping of algorithms to architectures that is the first requirement for analysis of execution behavior of algorithms for parallel computations.

To date, algorithm development has focused almost entirely on finding and exploiting the parallelism that exists in the set of algorithms that have been accepted as the standards. This work has been very valuable for obtaining relatively high performance from existing machines that exhibit parallelism. It has also had the positive effect of influencing the designs of some proposed parallel architectures. However, if we are to have any chance of reaching the full potential of parallelism, algorithm research must expand from its present rather narrow base.

In particular, more attention must be paid to algorithms whose serial complexity is not as good as the best serial algorithms, but which possess attractive features of parallelism that would permit them to execute faster in parallel mode than would parallel versions of the serial algorithms.

Another major areas for research is in asynchronous algorithms. With a few notable exceptions there have been no new algorithms proposed which take advantage of the fact that asynchronous computing is a reality. Thus asynchronous systems are forced to pay the price of frequent synchronization in order to execute the traditional algorithms.

Finally, it should be noted that improvements in algorithms, at least for numerical computation, have been as influential in improving computing speeds as improvements in hardware performance. Furthermore, since we are only beginning to study truly new parallel algorithms, we might expect significant rapid improvement from this area.

The enormous range of the spectrum of parallel architectures and the complexity of the mappings of real algorithms to realistic models of parallel architecture creates a need for automation of both representation of architectures and realization of mappings. The problem is akin to, and equally as complex as VLSI design.

There will not be pure architectural realization of models of computations. The interactions between models of computations, algorithms and actual architectures are illustrated in Figure 1.

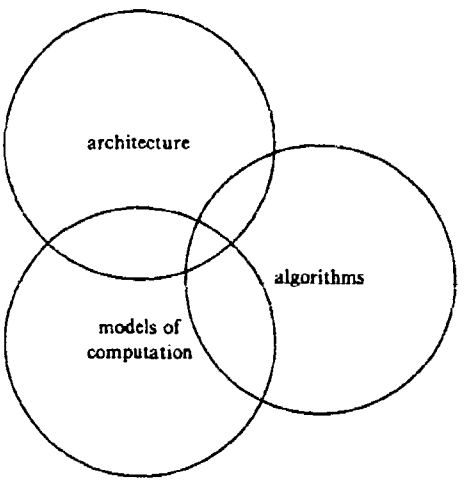


Figure 1:
Interactions of models, algorithms, architectures

These interactions must be taken into account if models of execution behavior are to be validly extendable to real systems. The systems aspects of algorithms, such as memory management and I/O structuring must be addressed in the models.

2.2 *Application-oriented Research*

It is clear that much research in parallel computing will be motivated by scientists who have applications whose solution requires the highest possible performance. In addition, it is expected that no single parallel computer will possess characteristics that are appropriate for this diversity of applications. This situation presents at least two opportunities for research: an application may be investigated to determine its inherent parallelism without regard for any particular computing system; or an application may be studied to determine how best to structure it for a particular parallel architecture. The first approach would lead naturally to the definition of an architecture, while the second could identify strengths and weaknesses of an existing architecture.

The applications themselves fall into two categories discussed below. In the first, the computation contains some real-time constraint which must be satisfied for the results to have value. In the second, the computations require resources beyond what is available on today's most powerful computer systems.

2.2.1 *Real-time Processing*

In real-time processing, available computing power seems never to be adequate. To achieve the required performance, analog devices are widely used today in spite of their low flexibility and reliability. It would be most desirable if these devices could be implemented by digital computers with sufficient performance. Digital radars or beam-formers typically call for billions of arithmetic operations per second. Realtime image processing of a 1000 x 1000 image requires a similar computation throughput. These examples show that computation requirements for real-time signal processing are typically orders of magnitude beyond the capabilities of today's computers.

Besides high throughput requirements, real-time digital systems often have additional constraints on their sizes and power consumptions. Therefore, high speed but power-consuming technologies such as ECL cannot be used to implement these systems. Highly parallel computers implemented with high density and lower power VLSI technologies are the only possible solution for these applications.

Computationally demanding applications are usually partitionable in the sense that there is a large number of subproblems that can be solved in parallel. Moreover, parallel computing with high regularity can often be devised, and as a result cost-effective special-purpose systems may be used. As hardware costs continue to decrease and CAD tools keep improving, the special-purpose approach will become increasingly important, particularly in the real-time area.

On the other hand, one should be warned that there are also many computationally demanding applications which are not suitable for special-purpose implementation. Application requirements are constantly under change. The size and accuracy requirement of a single problem can vary a great deal from instance to instance. Multiple problem decompositions may be required. The problem decomposition issue is non-trivial in general; a good scheme must deal with the issue of balancing computation and I/O.

2.2.2 *Modeling, Design and Simulation Applications*

Needs for high-speed computation are growing. Established applications include fission energy research, fusion energy research, petroleum engineering, aeronautical design, chemical catalysis processes and nuclear weapon design. Potential applications include robotics and artificial intelligence. In most of these applications the associated models consist of numerous software modules each containing thousands of lines of code. High speed in the execution of these applications necessitates efficiency not only within the modules but across them as well. Thus, research is needed in the implementation of full-scale models on parallel systems. Further, the need is not only for implementation but also for education. Parallel thinking is crucial and the requirement for it spans applications from expert systems to weapon design.

Two approaches can be taken for implementing a specific application. One is to select an existing architecture and seek changes in structure and algorithms that will result in

efficient execution on the chosen architecture. The second is to look for natural "structure" and "parallelism" in the application independent of existing architecture. The results derived could have significant impact on architectural design.

2.3 *Software Systems for Parallel Computing*

Just as parallel computing properly includes sequential computing as a special case, so does parallel computing extend the usual requirements for system software support. Particular research areas which must be given attention include programming languages, programming methodologies and operating systems.

2.3.1 *Programming Languages*

There is a wide spectrum of language possibilities for performance improvement through parallel execution. The factors affecting the choice of such a language include target architecture, ease of use, and compatibility with existing code. The extremes and midpoint of the spectrum can be characterized as follows:

1 Conventional sequential languages: Languages with vector operations (e.g., APL) can exploit parallel processing capability through simple compilation techniques. A major issue in need of addressing for vector architectures is that of conformability of logical and machine vector sizes. On the other hand, languages with inherently sequential semantic structure require an optimizing compiler which removes nonessential sequencing, mapping sequenced operations into parallel structures. A variety of difficult optimization problems remain for future research.

2 Hybrid Languages: Partly in an attempt to address the compatibility issue, and partly out of consideration for certain physical architectures, language research possibilities include those which either augment conventional languages with concurrent constructs, or embed them within a concurrently executable superstructure, for example, one used for physical configuration purposes.

3 Languages expressly intended for parallel computing: This category includes dataflow and application languages which, while not *requiring* parallel execution, do not inhibit it by making explicit sequencing an essential construct. Also included here are languages which are more specific as to parallelism, such as those in which communicating sequential processes play an essential role. Problems here lie in the areas of language design and of automating the assignment of processes to physical units.

2.3.2 *Parallel Programming Environments*

Parallel programming is significantly more complicated than sequential programming because (a) it is difficult to keep track of several simultaneously occurring events, (b) there are potentially complex (time-dependent) interactions of the system's components and, thus, elusive and nonrepeatable bugs may occur, (c) the sheer size of problems requiring parallel computers leads to complexity, and (d) parallel processors are often "attached" to other machines so their behavior is observed only indirectly. Therefore, it is crucial to provide adequate support to the programmer, to overcome the above difficulties.

Parallel programming environments should provide such support and are thus an important research area. The environment provides convenient, interactive access to the programming language compiler(s) and utility software (e.g., loaders, libraries) as well as facilities for testing programs and tracing execution. Although we can build on the experience of sequential programming environments, there are challenging problems to effectively supporting a parallel architecture. The most serious problem is managing large quantities of data, as would be encountered in supplying input to or analyzing output from a 1000 processor system, or in following a trace of the execution. Another challenge is to be sensitive to the requirements of a particular architecture: a vector machine requires a different kind of support than do other kinds of fine grain parallelism (e.g., data flow) or gross grain parallelism.

2.3.3 *Operating Systems*

Operating systems for parallel machines demand more than conventional sophistication. In addition to the usual issues, problems of physical load distribution, resource sharing, external communication and monitoring must be addressed. The interaction between processor and memory utilization becomes more subtle, particularly if multiprogramming is to be supported. The operating systems for many processor architectures must map from peripheral devices to processors and memories in extremely complex fashions. The software design problems are greatly inhibited by the absence of systems for experimentation.

2.4 *Parallel Architectures*

Research in parallel architectures relates principally to hardware organization and design issues. It is here that the various notions for achieving high-performance through parallel operations take on concrete hardware form. Clearly these forms interact heavily with the other research topics of this section. Parallel architectures represent the embodiment of models of parallel computation and the vehicles to execute solution algorithms (subsection 2.1). To be effective, parallel architectures must capture these models in such a manner that the languages used to express the parallel algorithms, and the associated basic software (i.e., compilers, utilities and operating systems) are supported efficiently (subsection 2.3). Furthermore, the architectures must have the resources (e.g., processing power, I/O bandwidth) to satisfy the requirements of the computational application being considered (subsection 2.2). This subsection discusses some of the principal research issues outstanding in parallel architectures.

2.4.1 *Architectural Styles*

Since architectures implement computation models, it is natural that there are about as many proposed architectures as there are models of parallel computation (see Section 2.1). Basic research questions here revolve around which model-architecture combination is best from the viewpoint of a given set of performance measures and application problems. In addition, within a particular model-architecture style, a host of research questions related to that style are typically present. These latter questions can have a significant effect on system performance. Unfortunately, there is still no comprehensive theory which permits one to characterize and classify the

various model-architecture styles which are possible. This is a fundamental research question which should continue to be addressed.

Two architectural styles which are frequently cited are SIMD and MIMD. An SIMD (Single Instruction stream, Multiple Data stream) machine typically consists of a control unit (to broadcast the instruction to be executed), a number of processors (each executing the same broadcasted instruction), a number of memories (to hold data), and an interconnection network (to provide communications among processors and memories). An MIMD (Multiple Instruction stream, Multiple Data stream) machine typically consists of a number of processors (each executing different instructions), a number of memories (to hold data and programs), and an interconnection network (for communications).

The MIMD style of parallelism differs from the SIMD style in that the processors in an MIMD system operate asynchronously with respect to each other, unlike the lock-step synchronous operation of the active processors in an SIMD machine. This results in the MIMD system having increased flexibility, however, at the cost of increased synchronization, task scheduling overhead and programming complexity.

The brief description above points to a number of important research questions concerning the design of such machines. Both machine types, for instance, require high performance interconnection networks if the communication costs associated with parallel processing are to be contained. This is of such importance it is considered separately in

2.4.2. In MIMD machines, problems of synchronization between processors become important as do problems related to system partitioning and reconfiguration. Fundamental questions remain as to whether shared memory, explicit message passing or some combination of these is the best approach to information exchange for a given application environment. Other key issues concerning design of pipelines, associative processors and pattern matching networks, and input/output data control (just to mention a few) continue to require the development of design and performance models. Alternative approaches to the development of subsystems can be evaluated with these component performance models and subsequently integrated into overall systems performance models (section 2.5).

While MIMD and SIMD represent two broad architectural approaches to achieving parallelism, there are many variations. Two of which show much promise are dataflow machines and systolic array architectures. Dataflow architectures follow directly from computational models based on dataflow graphs and applicative language representations. Systolic architectures follow directly from space and time representations of certain algorithms which map directly onto geometrically regular VLSI (Very Large Scale Integration) structures. Both exploit parallelism in ways which have promise for achieving high performance, and while research is being pursued here, significant gaps remain in our understanding of the limitations and most effective implementations of these architecture styles (in addition to numerous software issues).

Another interesting distinction which can be made between parallel architectures on a different dimension is general-purpose versus special-purpose systems. In a general-purpose system, parallel solution algorithms for a very wide range of applications can be found (although possibly not with the same effectiveness). Special-purpose architectures are typically more limited in application scope and are designed to perform best for one particular problem class, such as image processing, gate level simulation, or dynamic systems simulation. Such special-purpose machines may have extremely high performance in their selected problem domains, and their problem domains are often of such importance that development of such specialized machines is warranted. Fundamental research related into the tradeoffs associated with special- versus general-purpose parallel processors is needed to broaden our understanding of the essential performance constraints corresponding to particular parallel processor designs.

Finally, it is likely that no single architecture style will emerge as dominant over all application classes. Given that premise, there are interesting research possibilities in pursuing architectures which are designed to be dynamically reconfigurable. Such architectures could be configured by software to conform to the given problem, thus matching solution algorithm to architecture in a manner that results in best performance. Already some research along these lines is being pursued. Proposals have been made to build large-scale multiprocessor systems which can be dynamically partitioned to form one or more SIMD and/or MIMD virtual machines of varying sizes and allow the same processors to switch between the SIMD and MIMD modes of operation, all under software control. Similar motivations are behind ideas to build regular VLSI processing arrays whose connection paths are reconfigurable. Research into reconfigurable architectures has only just started and should be continued since it may be that true generality may only come by dynamically adjusting architecture to problem.

While we have made some progress, many fundamental questions remain, and we are now at the juncture where implementation experiments are essential. These fundamental questions should continue to be pursued as pure science investigations in parallel and in conjunction with experimental efforts.

2.4.2 *Internal Communications*

Within a large scale parallel computing system data must be shared and communicated among the processing units. Furthermore, the movement between secondary and primary memories of the large volumes of data on which parallel systems often operate must be done effectively so as not to negate or limit the advantages gained from parallelism.

The design of an efficient communications structure for transferring data among the multiplicity of processors and memories is an important area for study. Research issues include: choice of general structure (e.g., "direct" links versus "multistage" networks), choice of implementation style (e.g., circuit-switched versus packet switched), choice of topology (i.e., interconnection pattern), methods for "permuting" data (for SIMD operation), ways to perform one-to-many connections, designs to move different data items in the system simultaneously (i.e., high bandwidth), designs that can be partitioned for execution of different subtasks simultaneously and/or to allow multiple

simultaneous users, designs that can assume different configurations (i.e., support-system reconfiguration), and methods to efficiently control the routing of data through the network.

2.4.3 *Input/output Bottlenecks*

As parallel processing and computational capabilities increase, the current problems of limited input/output bandwidth will become worse. Providing gigaflop parallel machines with data in a fast, appropriate manner is a major task, and may become a significant bottleneck in achieving overall processor performance. Activities directed towards alleviating this problem should be encouraged. Much of the technology-related work such as providing higher speed mechanical and electronic disks will be pursued in industry in response to their ongoing needs in this area. This will likely not be enough, and research ideas in the areas of data organization and general input/output architectures which creatively use given bandwidths and devices in novel ways should be pursued. This of course will interact heavily with the research associated with general parallel architectures and interconnecting networks.

2.4.4 *Technology Design Issues*

The previous discussion has dealt principally with higher level architecture and design problems. Given a generic parallel architecture, an interconnection network and input/output scheme, a host of associated research and design issues arise when implementation is attempted. These issues fall between the component and higher architecture levels of design, and may have a profound effect on the modularity, scalability, fault tolerance and speed of the resulting system. Research issues here relate, for example, to designing for high bandwidth, modularity and other performance measures under a host of physical constraints (i.e., heat dissipation, area, pin limitations) related to a given packaging technology. General models incorporating such constraints into design procedures are lacking. Other design problems, such as the distribution of control signals, and the interaction of such design decisions with both physical constraints and higher level performance goals, are typically solved on an ad hoc basis. Design models, methodologies, and procedures are needed. Finally, to build very large parallel processors, in a cost-effective manner, will require heavy exploitation of current VLSI technologies. The impact of VLSI on parallel architectures and the best ways of partitioning architectures across the VLSI design domains remain open and important research questions.

2.4.5 *Fault Tolerance*

The issue of fault tolerance is very important for large-scale parallel architectures. The large number (e.g., thousands) of computing and input/output devices comprising a big parallel system greatly increases the probability of a failure somewhere in the system. Furthermore, the size and organizational complexity of future parallel systems makes finding faults extremely difficult. Thus, much research is needed in the areas of fault detection and location in large systems.

On the other hand, due to the multiplicity of resources in a parallel architecture, there is great potential for incorporating fault recovery schemes (from a hardware point of view). The ways in which such fault recovery schemes can be incorporated into systems and controlled efficiently are also important subjects for future study.

2.5 *Measurement and Evaluation*

High performance will actually be obtained from parallel architectures when the algorithms executed map efficiently to the architecture. Efficient mapping must be based on a thorough and detailed understanding of the resource requirements of the algorithms and the ability of an architecture to deliver resources. Research to establish both of these knowledge bases is required.

It is a truism that we do not understand execution behavior of most common algorithms on serial architectures. Mappings of algorithms to parallel architectures is a problem of extensive dimensionality and much greater complexity. Understanding of execution behavior on parallel architectures will require major experimental effort as well as the theoretical work described in Section 2.1.

An essential component of future research is to obtain data characterizing the execution of actual production codes from a spectrum of application areas on a variety of architectures. Models of significant execution behavior patterns can be extracted from this data. These models, which may be very faithful and detailed, can be validated by simulated execution on models of the architectures upon which the data was gathered. Then extrapolation to unrealized but potentially interesting and significant architecture can be made through simulated execution.

An important element of these studies will be identification of potential bottleneck, areas with respect to imbalances in rations of resource usage. The identification of these bottlenecks may also suggest possible solutions. For example, if the number of memory accesses per instruction become a bottleneck then means of making a data fetch have a higher utility value may become a goal for architecture research.

Communications requirements and the modes, rates and types of interactions between the processes executing parallel algorithms is another example of an area requiring measurement and characterization. The effectiveness of translations lying between algorithms and architecture is another high return target for measurement and characterization.

Underlying this entire problem area is a need for both hardware and software to integrate measurement as an intrinsic component of system functionality. Only when the need for evaluation is considered as an integral part of a system will measurement be accomplished at reasonable cost.

There should be a very strong interaction between measurement and evaluation and research on modeling and algorithms. Meaningful measurements and characterizations must be founded on the framework provided by models and abstractions.

3.0 *FACILITIES*

This section specifies and justifies facilities necessary for the pursuit of critical research problems of parallel computing. These facilities service one or more of the steps in the development cycle shown in Figure 3.1. The most critical problem for university research is that of feasibility prototyping of architectural proposals which have been evaluated as worthy of being developed through the

feasibility prototype stage.

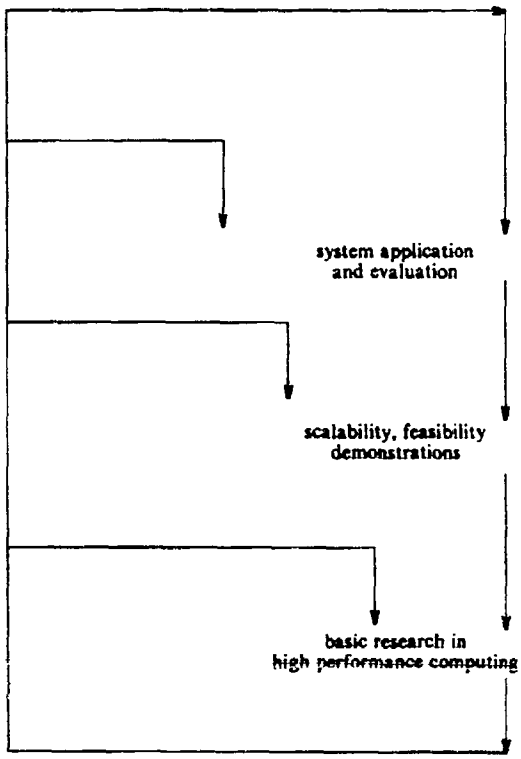


Figure 3.1 - Stages in Development of Very High Performance Computer Systems

Facilities for adequate simulations of architectures executing non-trivial workloads are beyond the reach of most university research groups. Such facilities would be an important element in expediting the evaluation of architectures.

It is the case that algorithms and software development could be greatly accelerated by providing suitable experimental access to the limited number of existing parallel systems. Examples include the Denelcor HEP and the Control Data Corporations's AFP. This work could materially shorten the interval to actual realization of effective utilization of parallel computing systems. It is also probable that such systems could help fulfill the need for simulation facilities as discussed in the next section.

3.1 *Simulation Facilities*

It is standard industrial practice to simulate designs of computers at all levels before committing the design to hardware. However, this practice is rarely followed in the university environment, primarily because of a lack of computing resources. An experiment which exercises all parts of a simulated design, or in which performance of the simulated system is measured under dynamic conditions, requires enormous computing power. The type of computing power required may depend upon the simulation experiment to be performed, but it always entails high instruction execution rate. In addition, large primary and secondary storage, fast arithmetic, and large address space are also often required. Since simulation programs are complex, their development can only be done in high level languages. Hence, the facilities for simulation must also provide support for program development.

3.2 *Support for Feasibility Prototyping*

We believe there is a critical need for the construction of prototypes of highly parallel architectures. While simulation and emulation are essential parts of the design process, only through a prototype can the design be verified sufficiently. Additionally, a prototype would provide potential users the means for evaluating the architectures and software, and for the experimentation with algorithms and languages which is critical to algorithms and software research. There are few commercial, highly parallel computers available today or projected to be available in the near future. This dearth of available architectural alternatives represents a serious impediment to further progress in a number of application areas of national interest.

Feasibility prototyping, in addition to being essential for significant results, also currently represents an unusual opportunity for success. Individual processor technology has reached the point where the engineering problems of implementing multiprocessor and parallel processor architectures are manageable. In fact, many vendors see an opportunity for commercial success of parallel structures given knowledge of which of the many possible parallel structures were 1) programmable and 2) computationally effective.

Universities have traditionally been good at proposing architectural ideas, implementing research software projects, and carrying out theoretical and experimental performance evaluations. On the other hand, they have been poor at technology innovation. While the silicon foundry approach makes it possible for them to produce new chips, they will not be able to exploit the state-of-the-art in performance, packaging or reliability.

It would seem that a great moment of opportunity is near in the possible wedding of high-speed architectures and VLSI components. But how can we produce a chip set that can be fabricated into high performance multiprocessors of various types?

An attack on the problem would seem at this moment to have a high probability of success for several reasons.

- Several universities and laboratories have well-developed efforts in architecture, software and algorithms for high-performance computers.
- Several companies have already produced high-performance component prototypes and are exploring the systems-level implications of these. The new manageability of the engineering problems of prototyping a parallel computing structure substantially lowers the traditional high risk that has been associated with such an undertaking and improves the chances of multiple successes. There are also several corporations that would be receptive to assisting and cooperating in the process of fabricating and debugging demonstration systems for truly promising designs. The promise in the design can be as much in the programming style or programming ease or programming effectiveness as it might be in unusual parallel architectures.

Prototype activities could take place in several ways. At lower funding levels, relatively small university teams could carry out the design process and with industrial help

for state-of-the-art problems such as board design, could construct a small working prototype. Basic software would also be provided by the university. This would result in sufficient hardware and software to permit limited experimentation by users. However, to carry these prototypes through to production might require significant redesign and additional software. A second method, requiring higher levels of funding, would require earlier (and more) participation by both users and industry. (For example, see Gajski, Kuck and Lawrie, "The role of government, industry, and universities in the production of very high speed computers", Appendix B.) This would result in earlier production of a final design, perhaps as quickly as five years, and significantly better experimentation facilities for users.

The eventual goal would be to have one or more very high-performance computers built with advanced architectures and technologies by industrial organizations that would be likely to continue product-oriented development of the systems. Certain key conditions should be met to ensure the success of such a venture. It would almost certainly make sense to carry out several parallel projects which would be of different natures and which could cross-fertilize each other. Each such project should follow the outline below:

• Prototype. The architecture should be worked out and a scaled down prototype built using off-the-shelf parts in a short-term (e.g., two- to three-year) project. By concentrating here on the architecture, the university would not be drawn into the fatal position of trying to push architecture and technology at the same time. Users and industrial partners would play a key advisory role in the initial design to insure the final design would be useful and manufacturable.

• Software and Performance Evaluation. In order to demonstrate the success of the project, software and algorithms should be produced by the university for the prototype so that it can be used on a variety of applications and its performance can be measured. Users would play a key role in evaluating the prototype and its software.

• Implementation. The initial design should have the property that it may be speeded up by several techniques.

a) The design might be architecturally scalable, e.g., large-scale replication of hardware should result in large-scale performance improvement.

b) The design should be technologically scalable, i.e., it should be amenable to cost/performance improvement through hardware enhancements by the industrial partner. For example, the use of fast circuits or the use of custom chips instead of gate arrays would allow a faster clock Custom VLSI chips would give a cost and size reduction as well.

The potential payoffs from such prototyping efforts are:

1 more rapid selection of the best ideas from parallel architectures and parallel programming,

2 demonstration of the step-function increase in high-speed computation that can meet pressing national needs,

3 demonstration of the directions necessary for industry to maintain world leadership in computing,

4 vehicles to promote and stimulate more advanced ideas in parallel computation,

5 and vehicles to improve the quantity and quality of graduate training in critical areas.

3.3 *Experimental Facilities for Parallel Computing*

The final measure of parallel computation will be its ability to deliver high-speed in support of real applications. If the ordering of processes inherent in serial computation is imposed in parallel computation then Amdahl's Law will be in effect. The result will be that small perturbations in algorithms, implementation, systems overhead, etc. may produce large perturbations in overall performance (e.g., performance will be quasi-stable). If sequential ordering is not imposed, then radically different analyses and algorithms must be developed. In either case, experimental computation will be needed to determine overall performance.

In order to carry out much of the research advocated in Section 2, it is imperative that the community have access to a small but diverse collection of parallel processing systems. In particular, the application-oriented research described in Section 2.2 will be effective only if computing facilities are available for evaluation of the concepts and approaches developed by that research. In addition, in order to develop the tools and techniques required for the measurement, modeling, and evaluation of systems discussed in Section 2.5, the researcher must have access to parallel computing systems.

In order to maximize the benefit of experimental facilities, they must be easily accessible from remote sites. In addition, they must be adequately supported in terms of people and hardware/software systems, so that they are stable and usable with a reasonable amount of effort.

APPENDIX A

STATE-OF-THE-ART IN PARALLEL COMPUTING

Parallel computing is not a new subject. Vannevar Bush noted the possibilities for parallel computing in his classic article. Parallel computing has been largely neglected as a practical means of increasing the power of computer systems because it was much simpler to increase the power of uniprocessors through technology enhancement. It is the end of this steady increase in power of uniprocessors that motivates current interest in development of multiprocessor systems. There has, however, been significant development of parallel computing over the past two decades. The history of parallel processing is given in anecdotal fashion in the first five papers of the Proceedings of the 1981 International Conference on Parallel Processing. This appendix gives an opinion on the state development of parallel computing. The organization of the survey is as follows:

- special purpose architectures
- general purpose architectures
- research project architectures
- algorithms and applications
- software and languages.

Special Purpose Architectures

The most successful parallel processing systems and also ~~perhaps the greatest promise for further short-term success~~ is in special-purpose architectures for signal processing and processing of sensor derived data. Goodyear Aerospace has been a leader here with the STARAN, Microcomputer Array Processor (MAP), and Massively Parallel Processor (MPP) systems. These special-purpose systems are all in production use. The PEPE system (another SIMD array) developed by Burroughs was also successful in its application of radar signal processing. The Control Data Advanced Flexible Processor (AFP) is the newest and perhaps most powerful representative of this class of systems.

The systolic array architectures proposed by Kung point the way to a new generation of special-purpose architectures. Prototype realizations of this architectural concept appear very promising.

There are major problems in the use of these special-purpose systems with respect to I/O systems and programming.

General Purpose Architectures

ILLIAC IV, an SIMD processor array with a square mesh interconnection structure, was the first large-scale attempt at a general-purpose parallel architecture. The fixed geometry of ILLIAC IV's interconnection network makes mapping of many problems and algorithms difficult and forced large data transfer overheads.

It is possible to configure multiple processors (usually 2 or 4 CPU's on a shared memory) of a number of vendors including IBM, UNIVAC, Honeywell and others. These processors cannot, in general, be easily programmed for close coupling of processes.

The Denelcor HEP is the only commercially available system which readily supports general-purpose parallel processing. The HEP actually multiplexes many processes on a single computation unit. It is an MIMD shared-memory architecture. The HEP has been demonstrated to be successful in parallel processing of a spectrum of numerical applications. The CDC AFP also has potential as an MIMD processor but lacks software. No significant applications have been made.

Research Project Architectures

There are many research projects either simulating or building parallel architectures. A list which is reasonably complete as of August 1983 is given in Appendix H of the Lax Report. Each project is exploring different methods for implementing communication and synchronization. There is a general consensus that a number of these conceptual architectures are well enough tested to merit production of feasibility prototypes.

Algorithms and Applications

It is only in recent years that serious attention has begun to be paid to parallel formulations of substantial problems. (An exception is the application work surrounding the ILLIAC IV project.) This lack of practical knowledge of how to use parallel architectures is likely to remain a long-term inhibiting factor for progress.

Most early work on algorithms was formulated in terms of abstract models of computation and is not directly applicable to actual realizable architectures.

Software and Languages

Operating systems for management of a rich and diverse spectrum of parallel architectures are almost untouched because of lack of motivation (the absence of actual systems).

The operating systems and distributed computing research communities have formulated languages for concurrent programming, but at first glance, these languages are poorly adapted for programming of close coupling of many simultaneous active processes.

The absence of software and languages is a severe barrier to progress, since without them, gaining practical experience is laborious in the extreme.

APPENDIX B

THE ROLE OF GOVERNMENT, INDUSTRY AND UNIVERSITIES IN THE PRODUCTION OF VERY HIGH-SPEED COMPUTERS

D.D. Gajski, D.J. Kuck, and D.H. Lawrie

Consider the goal of producing several types of broad-range, multi-gigaflop computer systems in the next decade. We will discuss some of the strengths and weaknesses of universities proceeding alone and companies proceeding alone. This leads to the conclusion that some kind of joint effort is the best approach.

Universities have traditionally been good at proposing architectural ideas, implementing research software projects, and carrying out theoretical and experimental performance evaluations. On the other hand, they have been poor at technology innovation. While the silicon foundry approach makes it possible for them to produce new chips, they will not be able to exploit the state-of-the-art in performance or reliability. Another recent trend in academic computer architecture research seems to be toward the one big idea approach to computer architecture. This seems to produce architectures with a few simple, easy to understand ideas, that lead to a lot of related theoretical work, but too often without enough real-world applicability.

The focus in companies is shifting somewhat these days. The traditional system houses are continuing in some cases with products in which they have a long-term investment (e.g., Cray, CDC), others seem to have sharply retreated (e.g., Burroughs), while some are trying to enter for the first time with new ideas (e.g., Denelcor). Some will attempt to copy the successes of others (e.g., the Japanese, Trilogy). An interesting shift of focus, however, is to the semiconductor houses themselves. Several are now producing interesting, innovative, low-speed VLSI products (e.g., the Intel 432). Several are breaking into the one megaflop (32-bit) chip or chip-set arena (e.g., TI and HP).

It would seem that a great moment of opportunity is near in the possible wedding of high-speed architectures and VLSI components. In fact, we have been approached by one semiconductor manufacturer with precisely the question, "How can we produce a chip set that OEM customers can fabricate into high-performance multiprocessors of various types?" It is clear that companies are frequently forced to announce new products without enough time to think through the architectural, software, and applications ramifications of their work. Preliminary evaluations of the Intel 432 seem to be rather mixed, indicating a product that was perhaps rushed to the marketplace too quickly. In the area of high-speed systems, it is probably even more difficult to get a set of components that will perform well in a variety of systems and applications.

A joint attack on the problem by industry/university cooperation would seem at this moment to have a high probability of success for several reasons:

1 Several universities have well-developed efforts in architecture, software, and algorithms for high-performance computers.

2 Several companies have already produced high-performance component prototypes and are exploring the systems-level implications of these.

3 The VHSIC program provides an example of a successful prototype program for this type of cooperation.

As indicated above, the chances for success seem low if university and company efforts proceed independently. However, by properly managing a joint effort, some rather important breakthroughs would be possible. In addition, since government laboratories are likely to be the first customers for very high-speed computers, representatives of these laboratories should be an integral part of the university/industry team from the beginning.

Goals

The goal would be to have one or more very high-performance computers built with advanced architectures and advanced technologies. Furthermore, these computers would be built by industrial organizations that would be likely to continue product-oriented development of the systems.

Certain key conditions should be met to ensure the success of such a joint venture. It would almost certainly make sense to carry out two or three parallel projects which would be of different natures and which could crossfertilize each other. Each such project should follow the outline below:

1 *Prototype*. The architecture should be worked out and a (scaled-down) prototype built using off-the-shelf parts in a short-term (e.g., two to three year) project. This milestone would include the development of register-transfer level diagrams together with timing diagrams that would be delivered to the industrial partner. By concentrating here on the architecture, the university partner would not be drawn into the fatal position of trying to push architecture and technology at the same time. Government and industrial partners would play a key advisory role in the initial design to insure the final design would be useful and manufacturable.

2 *Software and Performance Evaluation*. In order to demonstrate the success of the project, software and algorithms should be produced by the university for the prototype so that it can be used on a variety of applications and its performance can be measured. This milestone is a validation of the prototype and simulation of the final project. The government partners would play a key role in evaluating the prototype and its software.

3 *Implementation*. The initial design should have the property that it may be speeded up by several techniques.

a) The design should be *architecturally scalable*, e.g., large-scale replication of hardware should result in large-scale performance improvement. An example of a design which is not architecturally scalable would be one that depends on a single bus for interprocessor communication. While this approach might work for 4 to 8 processors, it

would clearly be insufficient for 1024 processors.

b) The design should be *technologically scalable*, i.e., it should be amenable to cost/performance improvement through hardware enhancements by the industrial partner. For example, the use of faster circuits or the use of custom chips instead of gate arrays would allow a faster clock. Custom VLSI chips would give a cost and size reduction as well. Again, a design which depends on a bus where delays are largely due to wire delays might not be technologically scalable.

The software developed by the university partner on the prototype should be able to run, with few changes, on the final implementation. Further software development by the university in parallel with the final product implementation should relieve the industrial partner of long software development delays.

Summary

A broad-based approach in which several government/industry/university teams are involved in separate projects would seem to be an ideal approach to reach the goal of multi-gigaflop systems in the 1980's. If the groups were competing with respect to system speed, quality of results and schedule (not budget dollars), they would also be able to contribute to one another by exchanging ideas, serving on joint advisory panels, etc.

Thus we believe that the team approach outlined above would result in maintaining this country's leadership role in high-performance computers, as well as providing the computation power necessary to maintain our leadership in other areas. It is a well-structured way to provide the initial support industry needs to undertake the production of a new and large computer product. At the same time, it provides significant training opportunities for graduate students in the beleaguered higher education system in this country.

INFORMATION SCIENCE

An Assessment of the State-of-the-Art and
Recommendations for Future Directions

Prepared for
NSF Information Technology Workshop
XEROX International Center
Leesburg, Virginia
January 5-7, 1983

by

Howard L. Resnikoff, Co-Chairman	Harvard University
Lotfi Zadeh, Co-Chairman	University of California, Berkeley
Richard J. Herrnstein	Harvard University
Michael E. Lesk	Bell Telephone Laboratories, Inc.
Roger Schank	Yale University
Henry A. Sowizral	The Rand Corporation
Joseph F. Traub	Columbia University
Donald Walker	SRI International
Martha E. Williams	University of Illinois, Urbana
Joe B. Wyatt	Vanderbilt University

INFORMATION SCIENCE

TABLE OF CONTENTS

1.0 *INTRODUCTION*

The rapid progress of computing and communications technology is changing the nature of life in the industrialized nations. As a recent report (Reference 1) on a related topic expressed it.

"Civilization is based on the interplay between mind and muscle. Since James Watt's perfection of the steam engine 200 years ago, technology has concentrated on supplementing and replacing human muscle power by the power of energy-intensive machines. In the coming century technology will surely concentrate on supplementing and, in the more routine contexts, replacing human mental activity by the power of information-intensive machines."

The force of this technological revolution has already set American society in motion, producing patterns of change in the economy, in national defense, in education, in how knowledge is discovered and communicated, in how people work and play, and in the relationship of government to the citizen. Conservative projections suggest that the pace of change will be sustained and possibly quickened in the next few decades.

This Workshop is concerned in general with the health of the scientific and engineering underpinnings of future advances in information technology. This is an issue of more than passing interest, and one to which attention cannot be deferred because the costs of missed opportunities are likely to be much greater than the cost of leadership: we are, willy-nilly, in a race with our economic competitors, with our military adversaries, and with time itself in a society that has begun but not completed a far-reaching process of change from old and well-known ways to less definite but surely far different new ones. For this reason, and for purely scientific and intellectual reasons as well, insuring the continued health and progress of information technology and the sciences that underlie it should assume a position of priority on the national agenda. The progress of information technology depends on developments in computer science and computer engineering, but it also depends on advances in information science. Indeed, as computers become more useful and more freely used by the general population, the importance of information science will increase correspondingly, for, amongst other things, information science is concerned with solving the problems inherent in information transfer between machines and the people who use them. Human information processing and machine information processing meet at this interface, where information processing problems that are independent of whether the processor is an organism or a machine assume priority and call for the cooperative efforts of researchers with expertise in both domains.

Many of the important research problems that are related to the more effective use of information technology will be solved by the private sector without stimulus or aid from government. We have tried to focus on important basic and applied research issues which are not likely to be pursued by the private sector.

In fields that are as rapidly changing as information technology, information science, and computer science, a research agenda based on short-term objectives is likely to be overtaken by developments. But in order to be of some value, an agenda must consist of problems that are not inaccessible to research methods now available or likely to become available very soon. We have kept these constraints in mind and have tried, where we could, to key basic long-term research issues to milestone special cases and experimental developments that can act as a guide and a test of progress toward deeper and longer-term goals.

The importance of experimentation in information science cannot be too highly stressed. As in all other sciences, the ability to test hypotheses and to explore uncharted regions of the parameter space in which phenomena can be represented is essential for healthy progress. But observation and experiment must be supported by a strong theoretical science if the discipline is to avoid the sterility of unmotivated and unguided data gathering.

Two topics of some importance have not found their place on the agenda of this Workshop. We believe that they

deserve attention as significant collateral areas of interest, and recommend that their role in the national agenda be given suitable consideration in the appropriate context. The first concerns the relationship between information and economics. There are important and difficult theoretical as well as practical issues that should be studied. As the economy increasingly is affected by the new information technology, these questions will assume an even greater practical significance than they do today. Certain aspects of this question have been addressed n a recent report to the National Science Foundation (Reference 2).

The second topic that deserves attention concerns research into the societal impact of information technology. Here we have no special suggestions, but would be remiss if we did not express our conviction that some attempt to gauge the societal consequences of this unprecedented technological change is called for. We are in agreement with the sentiments expressed in Section 9 of Reference 1.

2.0 *INFORMATION SCIENCE*

Information first began to assume an independent and quantifiable identity in the researches of physicists and psychologists in the second half of the nineteenth century. It grew in importance with the development of electrically based communications systems, and has emerged as a unifying and coherent scientific concept along with the invention and development of computers during the past 35 years.

The subject matter of a science of information is inherently abstract, for it concerns not the substance and forces of the physical world, but the arrangement of symbolic tokens or, as we may say, the structure of "patterns." Since symbolic tokens are necessarily represented by physical phenomena, as thoughts are represented by the electrical and biochemical states of the neural network, there is an intermixing of the physical sciences with the proper subject matter of information science. This makes it difficult to separate the pure properties of the latter from those of the former, which merely describe the physical embodiment of the informational patterns.

Until recently, the only known information processing systems were biological. Today they have been joined by electronic information processors based on computers and telecommunications links which are, to be sure, still quite primitive in comparison with their biological cousins, notwithstanding their pervasive and crucial role in all departments of civilization.

Information science studies that higher level of abstraction which encompasses both the "bioware" and the "hardware" implementations of information processing systems -- with what men and machines have in common -- rather than the problems that exemplify the differences between electronic and biological implementations of information processing functions. Thus, one main problem area of information science concerns the general measures and principles which we may expect will apply universally to information in all its manifestations, including the properties of patterned structures in nature itself.

In addition to studying the structure of patterns, a science of information must be concerned with the properties of information transfer. One relatively well-worked-out aspect of this problem falls under the heading of signal processing. Communication of information by real systems necessarily involves transfers of energy, so there is an irreducible interplay between the physical substratum which carries the information and the patterned structure which constitutes it.

The scientific study of information cannot be divorced from consideration of pairs of "inventories" or systems which are capable of bearing information. This is responsible for certain important "relativistic" features of the subject. In an everyday setting, one may think of a pair of people conversing, or of an information database and a user of information, as exemplifying the constituents of a pair; in the sciences, one will think of the information elicited from an experimental situation which involves the pair consisting of an experimental apparatus and an experimental observer. Regarding the latter, the role of the observer is well known to be critical in the realm of quantum phenomena. In a different but related way, it is also critical in information science.

The patterned structures about which the most is known are those associated with the processing of sensory signals by biological organisms, and especially by humans. Neuroscience has uncovered much of value about bioware implementations which limit and thereby partially define the processes which govern the representation and processing of information, and psychology and linguistics are revealing the processing aspect of the subject in a more nearly pure fashion. Thus, it is appropriate to consider information science from the standpoint of the cognitive sciences; some of these questions have been considered in Reference 3. But we should also recognize the limitations of psychology and linguistics as experimental sciences: although the initial conditions and externalities of the experimental arrangement can be modified by the experimenter, the structural properties of the system under investigation generally cannot be manipulated, nor can its internal state be prepared with any degree of certitude. This limitation does not necessarily deal a mortal blow to the role of experiment in the cognitive sciences: for example, astronomy, most ancient and exact of the sciences, is limited to observation alone without control of even the initial conditions of the observed system. But the types of observations that can be made in the cognitive sciences are limited, and this constrains the types of theories that can be readily tested by the experimenter. It is here that the role of the computer in information science is of crucial significance, for the computer scientist can prepare the internal state of the observed system as well as its initial conditions. For this reason the computer is the most important experimental tool in information science, and offers the promise of greatly accelerating progress in understanding the nature of information and the laws which govern it.

3.0 *FUNDAMENTAL RESEARCH PROBLEMS*

Several fundamental problems and general principles of information science have emerged from the preceding century and a half of research. The central problem concerns the nature of patterned structures. Within the realm of human information processing, this is realized in two particular problems:

3.1 *How Knowledge is Represented*

Understanding how knowledge is represented in the brain and how it could be represented in the machine, and the collateral questions of:

(1) How information provided by the sensory

systems is converted into the abstract forms which the brain actually uses, and

(2) The extent to which artificially constructed information-bearing forms of communication, such as language, mathematics, and music, reflect the internal organization and structure of knowledge and the (mental or machine) means for employing them.

3.2 *The Nature of Learning*

Understanding the nature of learning, with particular emphasis upon the relationship between the properties of the information to be learned and the internal state of the learner. This includes a characterization of the knowledge-base requirements of the (mental or machine) learner and processing rules as a function of the knowledge being acquired.

One general principle has emerged from studies in many fields, and research is needed to increase

3.3 *The Principle of Selective Omission of Information*

Understanding of the principle of selective omission of information, and its limitations. This principle is at work in all biological information processing systems. The sensory organs simplify and organize their inputs, supplying the higher processing centers with aggregated forms of information which to a considerable extent, predetermine the patterned structures which the higher centers can detect. The higher centers in their turn reduce the quantity of information which will be processed at later stages by further organization of the partly processed information into more abstract and universal forms so that the representatives of inputs to different sensory organs can be mixed with each other and with internally generated and symbolic information-bearing entities. This principle also governs the creation of "abstracts" and "indexes" of text information and it will surely play an important role in robotics and in artificial intelligence applications. This problem deserves more intensive investigation and there are indications that considerable progress can be expected in the short term in special areas.

There are at least three particular principles which appear to play important roles in this process of successive selective ommission and aggregation into more abstract forms of coded information. Each of these defines an important thematic direction for research. The first particular principle is concerned with

3.3.1 Investigating *how information processing systems can be adjusted to extremize the quantity of information relative to some "processing cost" constraints.*

The term "extremize" is equivalent to the phrase "maximize or minimize". Designers of computers and other information processing systems naturally attempt to maximize certain measures of information processed relative to the cost of processing it. But biological information processing systems and nature in general appear to be arranged so that information is extremized subject to constraints which depend upon the problem under consideration and are analogous to "costs".

The second particular principle concerns

3.3.2 Understanding *the invariance properties of information processing structures and measures* under appropriate group actions and other types of transformations.

Information processing systems and events often have natural symmetries associated with them. For example, a measure of the quantity of information gained from an observation should not depend on the unit of measurement marked on a measuring rod or instrument gauge, nor upon where the zero-point of the measurement scale is placed upon it. The arbitrary relocations of the zero-point and selection of the unit of measurement can be thought of as "symmetries" of the measuring process. They form an instance of the mathematical structure known as a "group", and it is the identification of the invariants of the groups that arise in information processing situations which is called for by the second principle. The use of group and other more general invariants is a powerful way of selectively omitting information.

The third particular principle is concerned with

3.3.3 Understanding the hierarchical and, more generally, the *optimally efficient organization of information processing structures.*

It has been long recognized that the introduction of a hierarchical organization generally increases efficiency. The assembly-line method of manufacture is one important economic example. The organization of governments, of military forces, and of large commercial firms tends to be hierarchical, because this is an efficient structure for the information processing elements of decision making. The architecture of contemporary computers, and especially of their memory systems, is largely hierarchical. Biological information processing can often be viewed as hierarchically organized. The relationship of short-term and long-term human memory offers one example. In humans, the relationship of foveal to peripheral vision, and in the echolocating bats, the relationship of the auditory fovea to the rest of the auditory receiver, provide striking examples of common goals achieved by abstractly similar hierarchical mechanisms functioning in grossly different physical circumstances.

The final basic research direction that is considered in this report is concerned with

3.3.4 Investigating *adaptive information processing strategies.*

Here we refer to information processing strategies that accommodate themselves to the circumstances of their use. Such information processing systems consist of collections of "generic rules" (rather than fixed instructions) which interactively modify themselves in the course of their application based upon the varying nature of the input data. Adaptive systems are inherently interactive and are essential in areas such as robotics, but adaptive strategies are even more fundamental in more general arenas of information processing where decision-making in the presence of uncertainty plays an essential role; in addition to decision-making in the "management" sense, this includes fundamental applications such as the machine-mediation of learning, and information retrieval and inference from complex data and knowledge bases. Current "expert" or "knowledge-based" systems offer primitive examples, largely based upon experimental *ad hoc* techniques, that

suggest the potential level of capability of the more sophisticated adaptive information processing systems that a systematic and appropriately funded research program could yield.

4.0 *SOME IMPORTANT APPLIED RESEARCH PROBLEMS*

Recent research has produced adequate algorithms for managing 1-dimensional data bases of ordered keys. Both B-trees and extensible hashing programs are efficient for this problem and packages for these methods are widely available. But information that is more vaguely specified, such as English or other natural language words, or pictures, are not so easily stored or searched with machines. Such data is becoming more and more important, however, as word processors, digital telephony, and graphics devices increase the supply of information in textual, vocal or pictorial form. In addition, these forms are the forms of data most commonly used by people, and thus the ability to handle them would greatly improve the user-friendliness of computer systems.

Various information science problems underlie the difficulties in this area. At present, we have difficulty matching people's ability to scan text for subjects; thus many people prefer to skim through a printed document rather than use an electronic form, even though delays are produced by waiting for the printed version. We have difficulty clustering or grouping information, and thus people ignore large piles of paper which they do not have time to sort through, and which cannot be sorted and routed by machine. Similarly, even those retrieval systems which do operate often flood the user with paper, and the inability to summarize what was found makes users waste substantial time. As the computerized systems continue to make it easy to produce information but do not help with its reading or delivery, we can expect that the "information flood" will increase and the amount of time wasted by individuals coping with it will become steadily larger. But the principal economic impact of the present situation is not measured by lost time, as significant as that may be for the highly-paid "information worker"; rather, it is measured by the opportunity cost that results from having failed to locate or identify relevant information produced by the information retrieval system but not provided by it in a usable form.

What, for example, can we do with the large amount of written text now available in machine-readable form? About 75% or 80% of the mail in a typical large office is internal, and since most companies now have word processing equipment, that material is available in machine readable form, and yet most companies do not yet have retrieval systems to store and forward this material automatically. In fact, at the present time, there are organizations where long-term storage depends on taking the machine readable form, printing it out, saving the paper, and discarding the computer-readable version. Basic research in knowledge representation techniques are needed to handle this rapidly growing practical problem. The research agenda should include investigation of:

- languages and structures for formalizing the content of documents so that they can be processed automatically;
- parsing and semantic analysis procedures for analyzing English text for storage;
- partial-match or other imprecise or fuzzy query handling.

Advances in the hardware area, such as associative memory and other innovative storage devices, to make online storage of large volumes of text cheaper and easier are needed.

Similarly, we can soon expect to have large quantities of human speech in digital form. At the present moment it is not possible to make the media conversion between text and speech (or between pictures of text and text) easily. Research in these areas is also needed and should include:

- speech recognition studies, drawing on the work of linguists and researchers in artificial intelligence as well as on information scientists (see the section of the full Workshop report concerned with artificial intelligence);
- signal processing developments, perhaps involving special purpose hardware (see the section of the full Workshop report concerned with parallel architecture);
- advances in decision theory and other areas involving complex matching problems (see Section 5 of this report concerned with adaptive information processing strategies).

Each incremental advance in speech recognition, although it may not be adequate for automatic dictation or full "command and control" communication with a machine, will nevertheless be useful and productive, as recent examples in simple machine control and various other elementary applications where computers must be used by novices or by those who do not have both hands free (e.g., in factory automation) is already beginning to demonstrate.

Picture or image processing is even more complex, and it is becoming more important as computer-assisted design and automated image acquisition systems are creating rapidly growing archives of image information in machinable form. There is a large background of literature in pattern recognition, but we do not yet have data bases in pictorial form. Such problems as the storage of maps, flowcharts, or other diagrams, or the viewgraphs so essential to modern bureaucracy, cannot be solved without advances in the underlying fundamental information science research problems. Intensified research is needed in:

- the analysis of multi-dimensional patterns, and the design of knowledge representation languages for such images;
- the ability to "abbreviate" or "summarize" features in pictures, and to transform and index them;
- coordination with projects in robotics and other areas depending on vision research and scene analysis.

For all of these problems, it is important to note that people and animals find them easy; this observation offers hope as well as a challenge to the information scientist. Scene analysis problems that a computer would find impossible (e.g., "search for a particular individual in this picture") can be solved by even a pigeon. Thus, it is not enough to be told that they are NP-complete, as interesting as such a result may be; practical ways of solving them exist all throughout the biological world. This suggests the potential value of a three-pronged attack: research into the fundamental problem and its properties, research into

computer implementations, and research into the behavior of organisms that can solve these problems; see Sections 5 and 6 of this report.

One overall theme that emerges from this discussion of concrete problems is the need to develop general tools for dealing with them, which in turn calls for more intensified research on the fundamental problems of information science.

The spread of computers beyond the professional computing community has had a dramatic impact on user interfaces. Programmers, as Eric Carlson has said, are people who are paid to put up with machines. As an increasing number of ordinary workers must interface with computers, the ability of the machines to accommodate to the people, rather than the need for the people to accommodate to machines, must be improved.

As computers interface with more people, they will have to work at the tasks in the way that people are used to and that is convenient for them. Many of these information processing tasks are very easy for people, but currently very hard for machines. Research is needed in the behavioral aspects of information transfer to reduce the impedance of the interface between people and information processing systems.

The previous remarks have concentrated on aspects of information retrieval and knowledge-based systems that concern user-friendliness and the incorporation of speech, imagery, and office-originated text materials into information processing systems. But during the past 20 years on-line databases have grown from a rare, specialized, and costly method of finding information to a universal, essential, and more affordable tool used by large numbers of professionals. Although past developments in the database field have demonstrated their importance, vitality, and effectiveness, the extension of database capabilities and their future development will be increasingly intertwined with research progress in certain key directions. We now turn to consider these themes.

The majority of the world's currently published literature and data resources are being captured in computer-readable form. They comprise an ever-growing set of databases that are publicly available through on-line/time-sharing systems. This is a relatively new phenomenon. Few databases are older than a decade, and it is only in the last decade that commercial on-line time-sharing services were developed and provided the public with access to large numbers of databases. Between 1975 and 1982 the number of word-oriented databases (bibliographic and textual, e.g. Chemical Abstract Services' "CA Search" and the New York Times' "Information Bank") has grown from about 300 to 900 and the number of records in those databases has grown even more rapidly from 50 million to 300 million. The use of those databases, through an increasing number of search service organizations, measured in terms of numbers of searches per year, has increased over the same period from 1 million to 8 million.

This growth is but one indication that we are living in the age of information and that people need and will use information available through computers. Even greater growth has occurred for numeric databases and modelling systems (e.g., Data Resources, Inc., etc.). The growth has been technology-dependent, but not technology driven. The need existed and was recognized previously but could not be filled until low cost terminals and communication networks made it technically possible and economically feasible to provide these information services.

Both the word-oriented databases and numeric-modelling databases have been used primarily in industry, government and academia for supporting research and decision making. Now, the potential for more widespread use of databases and databanks exists because computers have been introduced to the population at large and information resources are now accessible beyond industry, government, and academia. In the coming years, far more people will have the technological capability via their home computers and intelligent terminals to access databases and databanks to answer their queries.

Usage will no longer be restricted to the computer-literate and informatio-literate who cope with the poor image quality, multiplicity of protocols, command languages, system responses and messages, file loading difficulties, etc., that exist today. Information seekers need user-friendly systems that make these complexities transparent. Users want, and shall be able to simply ask a question and get an answer without going through the multiplicity of steps that are required by current retrieval systems and databases.

Users should be able to access all types of databases: word oriented, numeric/modelling, and graphic files. Interaction with systems should permit keyed alphanumeric input, audio input and graphic or image input as well as search and retrieval of digital, audio, and image or pictorial files and the different types of files should be interfaced with (or talk to) each other where the query solution demands. The selection and location of systems and databases – whether centralized or distributed in many separate locations – should be transparent to the user. It should be possible and convenient to effect retrieval without concern for the fact that databases with different file structures, different data representation, different vocabularies, coding, and notation schemes must interact with each other to provide a response that includes *only* the data/information required rather than an overload of information as is the case with today's systems.

Extensive research -- both basic and applied -- must be coordinated in order to pave the way for development of such transparent systems or, more likely, systems of systems, and in order to accommodate the inevitable growth in data resources. Very large databases are being generated but the architectures of computers are not geared for information retrieval and fast processing of large natural language databases. As has already been observed earlier in this section, the successes experienced in handling numeric data are not readily transferred to the natural language data of textual databases. If effective use is to be made of the tremendous quantities of full text information being generated in computer-readable form, research will be needed to pave the way for development of more effective means for: automatic parsing of natural lannguage, images and audio data; automatic disambiguation of homographs, automatic inference methods for discerning embedded information or implied information, etc. The development of such means cannot be accomplished on the current base of direct match and comparison techniques universally used in commercially available data retrieval systems.

Avoidance of information overload can only be effected by the development of improved techniques for selectively omitting information: for summarizing, abstracting, and extracting salient information from the vast and continually growing stores that constitute a major national knowledge base and data bank resource.

The potential for improving on-line and knowledge-based retrieval systems is great but significant improvements cannot be achieved without additional research. Some of the research will undoubtedly be conducted by the private sector where there are short-term payoffs but much of it will require more time for fruition and will be done only if it is funded by the government.

5.0 *SOME IMPORTANT BASIC RESEARCH PROBLEMS*

In this section, we discuss several trends in basic research in information science that include foundational issues in adaptive information processing systems exemplified by knowledge-based "expert" systems, mathematical theories of computation and decision making in the presence of partial information, and aspects of information processing that are common to biological and electronic information processing systems viewed from the perspective of the study of relatively simple biological pattern recognition and classification.

5.1 *The Role of Uncertainty in Information Science*

In their pioneering work on information theory, Shannon and Wiener postulated that information is related in a fundamental way to uncertainty, which in turn may be expressed as the entropy of underlying probability distributions.

The Shannon-Wiener point of view is certainly appropriate for problems in which the issues of interest center on communication of messages over noisy channels. In recent years, however, the emergence of experimental expert systems as illustrations of, and tools for investigating, an area of central importance within information science and technology has shifted the focus of attention from quantitative measures of information in messages transmitted over noisy channels to the demands of propositions which form the knowledge base of a question-answering system. For example, if an information item in a database is

"it is unlikely that there are significant deposits of uranium in northern Nevada,"

then it is important to have a correct interpretation of such an assertion considering that the terms *unlikely*, *significant* and *northern* do not have a sharply defined meaning.

In addition to the basic and pervasive issue of imprecision of meaning, there are the important issues of unreliability and incompleteness of data. More specifically, a proposition in a database may be reliable to a degree, say 0.8, on the scale from 0 to 1. Or, a datum which is of relevance to a query may not be in the database. This raises the fundamental issue of how to deal with information which is imprecise, incomplete or not totally reliable.

The available database systems do not come to grips with this basic problem. Thus, the standard assumption that is made is that whatever is in the database is precise and reliable, and thus if a datum is missing its value may be determined by a default rule. In expert systems, on the other hand, and especially in MYCIN and PROSPECTOR, there are facilities for dealing with unreliability through the use of so-called "certainty factors" which are basically the ratios of underlying probabilities, some supplied by experts and some by users. However, the rules of combination of evidence in MYCIN and PROSPECTOR are *ad hoc* in nature and rest on questionable assumptions concerning the conditional independence of evidence given the hypothesis. Furthermore, the rules of combination in the currently implemented system are not valid when the propositions which represent the evidence do not have a crisp denotation. For example, if X is a variable whose value is queried, then from the data, "X is not very large, with certainty factor b", one could not infer a tighter restriction on the possible values of X with a certainty factor c which is a function of a and b.

A possible approach to problems of this type is to use a combination of probabilistic and possibilistic techniques. In this approach, a proposition with an imprecise meaning is represented as a possibility distribution, and rules of combination of possibilities are employed to infer from data in which the underlying uncertainty is partly probabilistic and partly possibilistic in nature.

Clearly, the issues of imprecision, unreliability and incompleteness in knowledge bases are central to our ability to design adaptive information processing systems which have nontrivial advice-giving capability in the presence of uncertainty. This is particularly true in applications in which an incorrect decision may entail grave consequences, as in medical diagnosis systems, fault diagnosis and correction systems in nuclear reactor plants, emergency response systems in airplanes, etc. In such applications, it is essential to have a correct assessment of the reliability of a conclusion as well as a reliable analysis of the likely consequences of possible courses of action. In more general terms, this may be viewed as part of the basic problem of validation of the performance of an information system -- a problem which becomes quite complex when the response is qualified by a certainty factor.

At present, we have at best a limited understanding of how to deal with large databases in which the uncertainty of resident information is an important factor. It is essential to develop better theories of knowledge representation for this purpose and, furthermore, develop architectures which are tailored to the processing of large volumes of uncertain data without the use of high standards of precision.

In more concrete terms, some of the principal problem areas which relate to the issue of uncertainty in information systems and which are in need of further study and exploration within the next five to years are the following.

1. Processing of information in databases in which the data are granular, i.e., are sets rather than points.
2. Processing of information in language data bases in which the propositions are imprecise and, possibly, associated with probabilities or certainty factors.
3. Approximate inference from imprecise propositions and the development of systematic rules for combining non-independent bodies of evidence.
4. Development of techniques for validation and assessment of reliability of expert systems.

5.2 *Optimal Algorithms for Information Processing*

For most problems, only partial or approximate

information is known. Such problems can only be approximately solved, that is, one must live with uncertainty. In recent years, computer scientists have begun the task of creating a formal theory of computation in the presence of partial or approximate information which would have numerous applications including prediction, estimation, and control; distributed and parallel computation; scientific computation; statistical decision making; design of experiments; and mathematical economics, and which would have implications for the larger questions of applying adaptive strategies in pattern structure and learning. We cannot review all of the directions of this new work but the nature of one strand in this ambitious research program is conveyed by the work described below. Because there now seems to be an opportunity for significant advance, support for research on the mathematical foundations of information science should be maintained at a level that will stimulate the level of activity.

A vast number of papers have been written on optimal information and optimal algorithms for the approximate solution of particular problems. For example, the annotated bibliography of Reference 4 lists over 300 core books and papers devoted to this subject. Based upon recent progress, the time seems ripe to attempt to create general theories for the construction of optimal algorithms. We describe an example of such a theory.

The theory is information-centered. One indicates the problem to be solved and the type of information available. The theory delivers the optimal information and the optimal algorithm. It yields the problem complexity, that is, the minimal cost for solving the problem to within a certain level of uncertainty. It states conditions under which nonadaptive information is just as powerful as adaptive information. The theory is based upon the notion of "radius of information" which measures the intrinsic uncertainty of a problem if specified information is available.

The broad objectives of this theory are:

(1) To develop a framework and fundamental concepts so that researchers can think about optimal information and optimal algorithms in general.

(2) Currently algorithms are frequently obtained on an *ad hoc* basis. By using an information-centered approach, proponents of this theory believe that the synthesis and analysis of algorithms can be made into a science.

(3) To develop mathematical tools for obtaining optimal information and optimal algorithms.

(4) Obtain optimal algorithms for problems in a wide variety of application areas.

Two models have been studied:

(1) Worst-case normed model

(2) Worst-case model where uncertainty is measured without a norm.

The two models are presented in References 4,5; an expository account is given in Reference 6.

The theory of the "worst-case normed model" has only three basic quantities, an operator S and a set F which specify the problem and an operator N which specifies the information. N is called the *information operator*.

There exists an invariant, called the *radius of information*, which measures the intrinsic uncertainty in the answer. The radius of information depends on S, F, N but since S and F are typically fixed and uncertainty is studied as a function of N, the radius of information is generally considered as a function of N.

Despite the generality of the model, very powerful results have been obtained. In particular, two of the theorems suggest how optimal information and optimal algorithms should be defined; it turns out that optimal information minimizes the radius of information for all information of fixed cardinality.

Regarding the relationship between adaptive and nonadaptive information processing strategies, there are problems where it is known that adaptive strategies are intrinsically more powerful than nonadaptive ones. One interesting question is when nonadaptive information is just as powerful as adaptive information. Considerable progress has been made on this question. Just one result will be cited here. Let S be a linear operator and let F be a balanced and convex set. Let the information operator N be linear, that is, let it consist of n linear functionals. Then nonadaptive information is just as powerful as adaptive information.

The information-centered approach can be contrasted with the algorithm-centered approach, which is in widespread use. In this approach, an algorithm is obtained, often on the basis of *ad hoc* criteria. This algorithm is then analyzed. Then another algorithm is proposed and analyzed, and so on. In the information-centered approach, one merely states how accurately or completely a problem should be solved and indicates the *type* of information available. The theory then tells one the optimal information, the optimal algorithm, and the problem complexity, that is, how much it must cost to solve.

The models referred to in the previous paragraphs are both worst-case models. Average-case models must be investigated. These models are, technically, far more difficult to deal with than worst-case models. Preliminary results for the case that the information is exact indicates that for any "linear" problem (S and N linear operators) the average case is the same as the worst case! That is, both have the same optimal information, the same optimal algorithm, and, often, comparable complexity. A major investigation is needed for the case that the information is stochastic.

As indicated above, general results have been established for the worst-case models if S and N are linear. The following cases should be investigated: S nonlinear, average-case models, N stochastic.

This theory can be applied to distributed computation, which is becoming of increasing importance as parallel computers draw closer to the practical stage. Even problems capable of exact solution on a uniprocessor will be solved only under uncertainty in the distributed environments of the future since complete exact information on the current state of the distributed system will not be available. Distributed systems should be modelled. There are obvious similarities between distributed computation and decentralized economies studied in mathematical economics. It may turn out that these superficial similarities reflect more fundamental similarities which can be exploited so that results in one area can be utilized in the other.

Diverse areas, many of which are disciplines in their own right, should be investigated from the information-centered point of view. These include control theory, signal

processing (in this case it is particularly important to distinguish between traditional applications of Shannon's "information-theory" and the "information-centered" approach discussed here), and decision theory. The information-centered approach has recently been used to solve problems in estimation and prediction. A unified theory for the optimal solution of problems with partial or approximate information has a number of implications. It would provide an understanding of the relation between information and uncertainty. It would give problem solvers in many domains new and powerful tools. It is hoped that it will eventually provide a unified theory of information including information as used, for example, in control theory, information theory, decision theory, scientific computation, and economics.

5.3 *Biological Pattern Recognition and Classification*

Problems of decision-making in the face of uncertainty, of pattern classification and organization, of retrieval of information from knowledge bases, and of learning, all involve problems of categorization in an essential way. Indeed, categorization plays a central role in information science, at every stage in the operation of computing machines, and also in human concept formation and the structure of human knowledge. It is, therefore, appropriate that attention be devoted to categorization as a general information processing function, embracing its role as a general biological function. Since biological organisms at every level of complexity are still far more advanced than computing machines in their ability to form useful categories based on limited and noisy information, it will be well worthwhile to investigate the algorithms and processes they use from the more general standpoint of information science. Indeed, it is our belief that this approach, which combines methods and results from a number of disciplines, is highly promising and is likely to result in fundamental advances in our understanding of the general process of categorization.

It is sometimes assumed by students of human pattern recognition and its simulation that the interesting and challenging phenomena are found in human sensory systems uniquely. A characteristic attitude was expressed by Howard and Campion: "The human visual system is the only effective pattern classification system known" (Reference 8, p.32). Summarizing recent developments in cognitive psychology, a *New York Times* science writer echoed the attitude he must have heard in many interviews: "We human beings ... are concept-making creatures: Unlike any other animal, we have a natural ability to group objects or events together into categories" (Reference 9, p.48).

Biologists and psychologists studying stimulus discrimination often betray the complementary supposition about the animal subjects they observe either in laboratory experiments or in nature. In contrast to the complex and abstract patterns that human beings respond to, the nestling herring gull, for example, is said to gape for food, not at the sight of its parent as a whole, but simply in response to a small red patch on its parent's yellow beak. The inscrutably complex classification capacity of human beings is assumed to transcend vastly the corresponding capacities in animals.

In fact, recent evidence (reviewed in Reference 7) indicates that, at least under some conditions, categorization principals employed by animals are virtually as inscrutable and complex as those used by human beings. Pigeons, for example, have been trained to sort through photographs looking for instances of such categories as people in general or an individual person, trees, bodies of water, fish, letters of the alphabet, oak leaves, pigeons, and various regular geometrical figures, such as triangles or diamonds. Blue jays have performed comparably well looking for instances of cryptic moths; mynah birds, for instances of trees and people. Apes and monkeys have likewise mastered complex categories. A variety of subhuman species, including pigeons and monkeys, have learned relational concepts, such as distinguishing between symmetrical and asymmetrical designs, picking the match to a target color or figure or picking the nonmatch, and so on. In all of these studies, the categorizations in question have passed the test for generality with new stimuli. That is to say, after a pigeon, for example, has learned to distinguish between a given set of photographs containing trees from one not containing trees, it is tested with different photographs, to see if the pigeon's principle of categorization generalizes to new instances, as it must if it is at all comparable to those of human categorizations. As noted above, some degree of generalization was demonstrated by most subjects in each study, often to a high degree by all subjects.

As long as it seemed that human categorizations are uniquely complex, then the inability to simulate them seemed readily explainable as a corollary of the sheer complexity of the human brain, with its 10^{10} neurons each with multiple connections. The situation alters radically, however, in light of the discovery that animals with relatively small and simple brains, such as birds and perhaps fish, also perform these complex categorizations, and that they do so apparently with no greater difficulty than human beings do. Whatever the cognitive benefits of the enormous human brain are, they do not, at this point, seem to have much to do with categorization as such.

A classification problem is unavoidably a problem in establishing a principle or principles of invariance. Such principles pose more or less interesting problems in information science, depending on the degree to which they are self-evident from a simply physical inspection of the classificatory process. The more obscure the principle, the greater the challenge to analysis, and the greater the potential increment to an understanding of possible categorizing algorithms.

The existence of complex, multidimensional categorization by relatively simple organisms implies that evolutionary processes have discovered and implemented nontrivial algorithms far beyond any existing theory in information science, let alone its implementation. Research along this line might, to begin with, search for the lowest biological level at which present information science loses its ability to account for the observed categorization. At that point, there may be a reasonable expectation of discovering a biological solution to a complex classification task. Just as evolution presumably added classificatory power incrementally, so also may we expect an account of it to develop.

Another possible line of research is to search for the environmental factors that activate an organism's latent capacity to categorize. Presumably, categorizing, like any other behavior, is governed by the adaptive contingencies, the economics of activity. At some point in its round of

activities, an animal is impelled to categorize a stream of instances, notwithstanding the inevitable loss of information and the psychic costs entailed by doing so. How the economies of categorizing operate at the level of simple behavior may illuminate the comparable issues in human, and social, behavior, and provide guidelines for their effective implementation in machine information processing systems.

Nontrivial powers of categorization in subhuman animals provide a rare opportunity for research into the general problem that supplements traditional computer science approaches in a powerful way. With animals as subjects, experiments can exploit the potentialities of far greater behavioral control than they could with human subjects, and also the potentialities of physiological intervention. By studying relatively small nervous systems mediating categorization, it may be far easier to discover categorization algorithms than by studying the human nervous system with its vast capacities beyond mere categorization, or by performing computational experiments divorced from the realities of the biological evolution of successful categorization algorithms.

6.0 *EXPERIMENTAL INFORMATION SCIENCE*

Given the perspective on basic and applied research problems in the previous parts of this report, it is evident that information science is at a stage where it could greatly benefit from a set of research facilities that would sustain experiments in the field. We describe two quite different kinds of facilities in this report. The first focuses on experimental verification of theoretical predictions, reflecting the discussion, presented above, of the centrality, over a broad range of biological and physical forms, of the principles we see as underlying information science. Such a facility would be comparable to the use of particle accelerators in physics to verify theoretical predictions and explore heretofore inaccessible regions of the space of phenomena. The second concentrates on the organization and use, by human subject matter specialists, of information in relation to the increasing number of what might best be called knowledge resources. We consider each in turn.

It was observed in the previous section, concerned with basic research problems in information science, that even relatively primitive biological information processing systems possess categorization capabilities that far transcend those of any machine-based system available today. Moreover, recent work suggests that the pattern classification and categorization systems used by biological organisms are universal in the sense that they classify data into similar categories, and they are simple in the sense that they can be implemented in small processors (by biological standards); indeed, after accounting for the processing requirements for flight control and general physiological functions, there may be as few as several hundred thousand neurons available to the pigeon for its categorizing processes. A machine of this relative simplicity is within the scope of current computer engineering practice. This is not to suggest that a computer "copy" of the pigeon processor should be designed, but only that the pigeon's capabilities demonstrate that a universal pattern classification algorithm of substantial power and generality can be packaged in a device which is accessible in principle to current engineering and manufacturing techniques. The problem is, of course, that we do not know the algorithm -- we do not know how the pigeon performs its categorization tasks -- although we can be assured that at least one relatively simple algorithm exists.

We do, however, know something about this algorithm: it is necessarily *highly parallel*. Indeed, a distinguishing characteristic of information processing by organisms is that the tasks on which biological systems perform best -- those related to categorization, pattern recognition and classification, and learning -- are also the ones that utilize highly parallel procedures. Vision systems provide a striking illustration of this point. It is unlikely that important progress in machine categorization and pattern analysis will be made unless researchers can be guided by experiment, but current experimental investigations use available computing machines, and the machines are not adequate for investigations of many of these problems.

It is very important that such algorithms be discovered and, by combining the knowledge and methods of several related disciplines, it may not be possible to unravel their nature by a combination of theoretical and experimental research. The experimental research would be of two types, involving both extensive computer simulation of categorization algorithms of a complexity much greater than most that are generally considered at the present time (although they are, in an absolute sense, simple), and interdisciplinary experimental work designed to determine the critical elements of biological categorization algorithms (and thereby partially guide the machine experiments) by means of a systematic investigation of the abilities of lower organisms undertaken by consortia of psychologists, information scientists, mathematicians, biologists, computer scientists, and computer engineers.

6.1 *Center for The Study of Parallel Information Processing Systems*

We envision the establishment of a "CENTER FOR THE STUDY OF PARALLEL INFORMATION PROCESSING SYSTEMS" that would provide facilities for both theoretical and experimental work. The experimental effort would focus on the study of parallel bioware information processing, especially targeted on the key problems of categorization, classification, and pattern structure (since the experimental aspects of parallel hardware computers are already being addressed by university and private-sector scientists, and are the subject of another report of this Workshop) but also concerned with computing systems that might combine electronic hardware and tissue bioware components. The CENTER (which might consist of several physical facilities located in different regions of the country, each concentrating on an aspect of the effort) would provide opportunities for academic scientists to visit for periods of varying duration; it would have the necessary computing resources, appropriate animal management facilities for experimental work, and a small permanent technical support staff. The CENTER would make possible a level of cooperative and interdisciplinary research on the categorization problem and on other problems where electronic and biological information processing converges which cannot be undertaken anywhere in the United States today.

We recommend that the National Science Foundation commission a six-month study to determine the most appropriate organizational structure, geographical location(s), and funding level for such a CENTER to

provide the critical mass and experimental resources for an effective attack on the key problem of categorization, and furthermore, that the Foundation give high priority to implementing the recommendations of that study. This is an area of particular promise and opportunity.

6.2 *Knowledge Resources Facility*

An experimental facility which we will call a "KNOWLEDGE RESOURCES FACILITY" also is needed to aid researchers in the exploration of knowledge resources, ranging from databases of text and images to knowledge bases that are beginning to play an important role in adaptive information processing strategies.

Understanding the issue of experimental facilities as they relate to knowledge resources can best be done in the context of actual use. Let us consider, as the paradigm case, a scientist or scholar who is an expert in some field of endeavor (e.g., a doctor, lawyer, scientist, or humanist). This expert needs some information to solve a problem, test an hypothesis, answer a question. The data potentially relevant for satisfying that need are accessible in machine-readable form. The determination of what items are actually relevant can only be made by the expert. Given such a context, the issue to be considered is how information science can support such a user.

Equally important, however, is what can be learned as a result of providing such support. Information systems can be both tools for users and a context for research.[1] In particular, we are proposing, as an experimental test-bed for information science, to develop a system that will allow us, in the context of a human activity of central social importance, to address the four major topical areas set out at the beginning of this section of the report.

There are three basic kinds of knowledge resources appropriate for the

information systems being considered here:

(1) Primary data - texts, documents, and other files that pertain to the area of expertise. These resources would consist of the basic materials a user would expect to be available, although it may be necessary to provide some nonstandard items to allow for serendipitous influences. The user should not need to know anything about the way the database is structured in order to be able to access it.

(2) Secondary data - index terms, citations, abstracts, annotations, and commentary. These resources provide characterizations, at a metatextual level, of the content of the primary data. They can be used both to access the primary data and to embody evaluations of their content. Secondary data can be provided by users and programs; they should be labeled clearly to identify their source.

(3) Tertiary data - dictionaries (general and special purpose), registers of people, places, and things that contain biographical, geographical, and basic reference materials. They can be accessed by users and programs to process primary data or to facilitate the preparation of new items.

A system of the kind proposed must also have facilities that make it possible to organize the information retrieved, to prepare syntheses and new documents, and to interactively modify the range of resources specified above. Natural language dialogue facilities can be extremely useful for interacting with the system, since the initial form of a hypothesis or question is rarely the one that adequately satisfies a need for information. To the extent that a natural language interface can capture the language habits of its users, including jargon and usages, it can be much easier to assimilate than a special query language. In addition, natural language provides a compact way of expressing complex relationships. Where it is imprecise or ambiguous, it may be valuable, in the context of interaction with the system, for refining concepts or prompting the recognition of alternative interpretations.

Note that in the system being described the database is cumulative and that it progressively comes to embody the experiences of its users. At the same time, the original form of the primary data is retained so those items can be reanalyzed and reinterpreted at any time.

Now consider having the databases cumulative in another sense. No field is static; new primary data items are continually being created, and they need to be absorbed and accommodated within the system. Accordingly, procedures must be available for spontaneously initiating a set of processing tasks with each new data entry. Actually, some subset of those procedures is necessary to accomodate, in a similar fashion, the annotations and commentary entered as secondary data.

It should be obvious on reflection that the system being described provides a context for research on (1) the categorization, classification, and comparison tasks associated with invariance; (2) the summarization and abstraction tasks associated with selective omission; (3) the representation tasks associated with optimal information processing; and (4) the interference and learning tasks associated with adaptive information processing. It also should be clear that the initial form of such a system will only provide partial solutions to those problem areas. Nevertheless, the feedback associated with the use of the system by experts who really need the information it contains is essential as a guide for successive revisions.

Given the foregoing prospective, it is appropriate to address the resource requirements for the development of a "system for experts" of the kind we have described. One of the major problems in its implementation has been the lack of hardware facilities with an address space large enough to accommodate the large databases that are entailed. With the advent of the LISP machines and other computers with capacities of 2^{36} bits, it is possible to proceed.

Increasing amounts of textual data are becoming available in computer-readable forms as by-products of news wire services and computer-directed phototypesetting, and optical character recognition is now a cost-effective technology. Dictionaries and other documents classed above as tertiary data are also computer readable. A substantial

1 Although the resources considered here could actually be used by any person who has a need for information and can tell when that need is satisfied, the expert can best provide the kind of feedback required to guide modifications and extensions of the system.

amount of research will be required to know how best to manage these latter items but initial work has already begun. Eventually, these reference works that embody would knowledge can be used in language understanding programs both inside of the system we have been describing and for other purposes associated with command, control, and decision making. They can also serve, particularly within the proposed context of research, for establishing a substantive base for research on expert systems.

The system for experts we have been describing will require the coordinated efforts of people from information science, computational linguistics, and artificial intelligence. It will also require the dedicated subject matter specialists, the experts for whom we are developing the system and whose use will provide us a context for exploring a broad range of research issues in information science.

7.0 *EMPLOYMENT, MANPOWER, AND CURRICULUM ISSUES*

Information technology is contributing substantially to an employment crisis in the United States that will exist, perhaps even increase in complexity and intensity, well past 1990. The phenomena resulting from this crisis range from the displacement of skilled and semi-skilled blue-collar workers by robots to the serious shortage of PhD-level scientists and engineers available for research, development, and teaching in fields related to information technology. Virtually all parts of the U.S. economy are affected by this crisis, although some appear to be more seriously troubled than others.

A recent study by Kent Curtis of the NSF concludes that "...there is a shortage of computer manpower which is expected to persist for the forseeable future but society is responding, perhaps as rapidly as possible, to provide the trained people required by business, industry, and government." After this hopeful conclusion, Curtis goes on to say "only the educational institutions of the U.S. have what might be described as a crisis, a staffing peoblem which seems to have no solution with the context of normal supply/demand forces."

Other studies have not been so optimistic about business, industry and government's collective ability to provide for themselves in this key area. Indeed, one recent study estimated an annual shortfall of 50,000 "computer-trained" scientists and engineers at the bachelor's and higher levels persisting past 1990. Taken together, there is no doubt that a shortage exists, and it appears that business, industry and government are satisfying some of their current needs by hiring people directly from universities and by hiring young people at the baccalaureate level who would otherwise have continued to complete the PhD for a career in education and research. At the same time, the number of undergraduate majors in fields relating to information technology is increasing dramatically as is the number of non-majors at all levels who take courses in computer science and related fields. These demographics result in sharply increased demand for both faculty and technological resources for colleges and universities.

However, the problem is not restricted solely to colleges and universities. The demand for introductory courses in computer science in the secondary schools, coupled with the desirability of using computers in teaching mathematics, physics and other courses, has caused a crisis in teacher recruiting and teacher training that is being met even much less successfully than at the college level.

In addition, there are the new employment opportunities that exist as a result of the technology itself. The number of "information professionals" employed in the United States in 1980 exceeded 1.64 million (Reference 10). The current annual growth rate in employment in this section has been estimated to exceed ten percent. At least half of these positions are related more to "information" than to "technology," including a range from database management to audio/visual/interactive media specialists.

Finally, there are all of those jobs affected by technology in sufficient measure to call for retraining, at a minimum -- replacement at worst. The American automobile industry is among the most visible examples of the potential negative effects when technological innovation is managed poorly. *USA TODAY* is a visible example of a new business venture and employer that is built on innovation in information technology. There is no doubt that less costly and more sophisticated information technology will affect other areas of business and industry in equally dramatic and significant ways.

Even with this brief summary of a very broad set of issues that link technology and employment, some issues emerge that are appropriate for federal intervention and can be addressed by NSF support:

A. Grants-in-Aid (Merit Scholarships) to outstanding students for graduate study in science and engineering fields related to information technology.

B. Post-doctoral fellowships in science and engineering fields related to information technology.

C. Institutional matching grants for equipment and other technological resources (perhaps coupled with other incentives for business to provide technological resources to educational institutions).

D. Support for research in the cognitive aspects of information science (see Reference 3, pp. 1-8).

E. Support for research in information technology that is important to the national interest, but is unlikely to be funded by business and industry directly or by other parts of the Federal government (see Reference 1, pp. 16-24).

F. Research on the economics of information (see Reference 2, pp. 16-24).

G. Research on and development of economic indicators and models to measure and forecast trends in the "technology economy" with special emphasis on the relatively neglected parts of the economy dependent upon information technology.

H. Support for the design, development, and experimental testing of educational materials using new information technology-based delivery systems for the fields most directly affected by information technology.

8.0 *PRIORITIES*

The problem of determining priorities in a rapidly developing field of science is a difficult one, and it is also one likely to produce as many opinions as there are respondees. For these reasons, this Working Group believes that it is unwise to attempt to specify priorities in detail; this is a task better left to the professional Program Managers at the National Science Foundation who have an unparalleled view of the field and its cognate disciplines. At the general level, however, there are some observations about

priorities that we believe have the support of the community of information science researchers.

1 The most important issue where the National Science Foundation can play a role is that of providing sufficient funds for research in information science. The theoretical information science and behavioral aspects of information transfer programs of the Division of Information Science and Technology at NSF have not fared well in recent years, although the quality of these programs is high. The Division's budget has lagged behind those of cognate fields, and the funds available to it are clearly insufficient to meet the need. During the past 5 years the budget has not kept pace with inflation and today it is at so low a level that it can only be considered marginal. In 1977 it was recommended that an appropriate level of funding was $10 millions (1977 dollars); the current budget is less than $6 millions (1983 dollars).

2 Information science has reached that stage of development where experimental facilities have become essential, and developments in microelectronics and computer engineering have made it possible to provide appropriate facilities with the large storage capacities and high processing rates that are needed. This report proposes two experimental facilities. Both are high priority. The proposed CENTER FOR THE STUDY OF HIGHLY PARALLEL INFORMATION PROCESSING SYSTEMS is likely to have the greater scientific payoff, while the proposed KNOWLEDGE RESOURCES FACILITY is likely to have more immediate practical consequences.

3 Development of improved curricular materials for a university concentration in information science.

4 Support for theoretical information science should be increased.

5 Support for research into the behavioral aspects of information transfer should be increased.

References

1 "Report of the Working Group on Information Technology", *Research Opportunities in Information Science and Technology*, National Science Foundation, NSF 82-63 (1982), pp.9-15.

2 "Report of the Working Group on the Current Status of the Interface between Information Science and Economics", *Research Opportunities in Information Science and Technology*, National Science Foundation, NSF 82-63 (1982), pp. 16-27.

3 "Report of the Working Group on Behavioral and Linguistic Research Bearing on Information Science", *Research Opportunities in Information Science and Technology*, National Science Foundation, NSF 82-63 (1982), pp. 1-8.

4 Traub, J. F. and Wozniakowski, H.: *A General Theory of Optimal Algorithms*, Academic Press, 1980.

5 Traub, J. F., Wasilkowski, G., Wozniakowski, H.: *Information, Uncertainty, Complexity*, Addison-Wesley, 1983.

6 Traub, J. F. and Wozniakowski, H.: "Information and Computation", to appear, *Advances in Computers* 23, Academic Press, 1980.

7 Herrnstein, R.J.: "Stimuli and the texture of experience", *Neuroscience and Biobehavioral Reviews*, 6, 1982, 105-117.

8 Howard, J. M. and Campion, R. C.: "A matrix for pattern discrimination performance", *IEEE Transactions on Systems, Man, and Cybernetics*, SML-8, 1978, 32-37.

9 Hunt, M.: "How the mind works", *The New York Times Magazine*, 24 January 1982, 30 pp. 10. "Final Report on the Manpower Requirements for Scientific and Technical Communication: An Occupational Survey of Information Professionals", National Science Foundation Project DGI-7727115 Report (June 30, 1980).

RESEARCH IN SUPPORT OF SOME MAJOR USE AREAS

An Assessment of the State-of-the-Art and
Recommendations for Future Directions

Prepared for
NSF Information Technology Workshop
Xerox International Center
Leesburg, Virginia

January 5-7, 1983

by

Floyd H. Hollister, Chairman	Texas Instruments, Inc.
Gerald Estrin	University of California, Los Angeles
K.S. Fu	Purdue University
Robert J. Hocken	National Bureau of Standards
W. Frank King	IBM
Daniel P. Loucks	Cornell University
James H. Mulligan, Jr.	University of California, Irvine
Philip N. Nanzetta	National Bureau of Standards
Alfred M. Pietrasanta	IBM, Systems Research Institute
Robert J. Spinrad	Xerox Corporation

RESEARCH IN SUPPORT OF SOME MAJOR USE AREAS

TABLE OF CONTENTS

1.0 *INTRODUCTION*

This subgroup was formed for the purpose of assessing computer science, computer engineering, and information science from the viewpoint of those who are attempting to apply them. Our objective was to determine what additional research might be required.

Clearly, a subgroup of eight or nine people cannot represent adequately all areas of use. Our intention was to explore this subgroup's interests and determine whether productive research recommendations might evolve from consideration of such a restricted sample space of applications. If so, NSF could assess whether or not to expand this kind of activity.

In choosing the use-areas to be addressed, we wished to represent two quite different kinds of user communities: those likely to be already tightly coupled to the computer science, computer engineering, and information science research communities and those which were likely not to be.

Our choices were:

Pattern Recognition and Image Processing, as likely to be tightly coupled to computer engineering;

Software Productivity and Office Automation, as likely to be tightly coupled to computer and information science; and

Large-Scale Systems, including CAD/CAM, Automated Production, and Civil Engineering Design Aids, as likely to be loosely coupled to all areas of our research interest.

The group also discussed issues related to Manpower/Resource Requirements and Interface/Industrial Requirements.

As the following individual reports on these areas of use will reflect, those areas which are tightly coupled to computer science, computer engineering, or information science have been able to make effective use of the technology resulting from their respective research and are able to make specific and concrete recommendations concerning additional research required. Those which have been loosely coupled have had considerable difficulty in effectively employing these research results in their applications.

Although our sample size is not statistically significant, it appears that there is a genuine problem in those areas which are loosely coupled and that there are steps which NSF and DOD can take to overcome these problems. Specific recommendations are made in the report below which address research to support the design of large-scale systems employing computers.

2.0 *COMPUTER ENGINEERING RESEARCH IN SUPPORT OF PATTERN RECOGNITION AND IMAGE PROCESSING*

2.1 *Scope of Consideration*

The subgroup adopted the view that image processing includes all multidimensional processes which require the manipulation of arrays of the form I(X1,X2...XN). Thus, such arrays generated by seismic, infrared, visible, ultraviolet, acoustic, radar, NMR and other techniques were included as images.

In accordance with the position paper, the subgroup focused its considerations primarily on the hardware and associated computer engineering required to process image data in digital form. No specific attention was given to the algorithms or software required to process images. This related and important area should also be given consideration.

2.2 *Importance of the Application Area*

There was general agreement among subgroup members that having cost-effective means for processing image data in real time is essential in a number of areas of national importance.

Important defense applications requiring pattern recognition image processing (PRIP) include:

- Processing multi-spectral and radar satellite surveillance data
- Processing visible, infrared and radar images for navigation and guidance of tactical and strategic weapons.
- Processing acoustic data for detecting and tracking submarines or guiding anti-submarine weapons
- Processing digital map data
- Guidance of robotic weapon systems.

Important industrial applications include:

- Processing images for automatic inspection
- Control of robots used in automated manufacturing and assembly
- Automated security systems
- Processing seismic, magnetic, visible, infrared, ultraviolet, and radar imagery for detecting and localizing minerals and hydrocarbon deposits.

Important medical applications include:

- Enhancement and interpretation of X rays
- Generation and processing of NMR images
- Generation and processing of sonograms
- Processing of images for analyzing blood, tissue and similar samples.

Other important applications include:

- Monitoring pollution
- Assessing agricultural conditions
- Detecting forest fires
- Determining and forecasting weather
- Office automation.

2.3 *Computation Requirements*

The computational demands imposed by processing images in the applications described above can easily generate the need to perform millions of floating-point operations per second (FLOPS). For example, applying a simple 3 x 3 neighborhood operator in preprocessing a 512 x 512 pixel image thirty times each second can generate the need to perform more than one hundred million FLOPS. If each pixel in such an image contains two bytes of information, then more than one-half million bytes of memory are required to store it. Demands of this magnitude are commonly encountered in the applications described above. Some of the applications can generate demands orders of magnitude in excess of these. One member of the subgroup estimates that an adequate processor for performing pattern recognition and image processing (PRIP) operators will have to perform at least 100M FLOPS with a memory bandwidth of at least 256M bytes/second in the 1980's and that these demands will further increase by an order of magnitude in the 1990's.

2.4 *Computer Architectures for Pattern Recognition and Image Processing*

Conventional single instruction single data (SISD) stream computers are designed primarily to process one-dimensional strings of alphanumerical data. To process multidimensional information on SISD computers requires image coding and picture transformation such as projection and registration. Sequential machines cannot efficiently exploit the parallelism inherent in most pattern recognition and image processing (PRIP) operations. On the other hand, large parallel computers, such as single instruction, multi-data (SIMD) stream array processors and multiinstruction, multi data (MIMD) stream multiprocessors, are not necessarily cost-effective in implementing simple and repetitive image operations over very large and, sometimes, dynamically changing image databases.

The architectural and functional features below focus on the interplay between computer architectures and PRIP applications. In general, a PRIP computer should have as many of the following capabilities as possible:

1 To explore spatial parallelism, a pattern-analysis computer can be equipped with replicated arithmetic/logic units operating synchronously and utilizing a high degree of pipelining;

2 Some PRIP computers choose a multiprocessor

configuration to support synchronous computations in MIMD mode. Data flow multiprocessor systems have also been suggested for PRIP computations;

3 An appropriately organized memory system is needed to store and manipulate images. Fast and intelligent I/O and sensing devices are needed for interactive pattern analysis and image query processing;

4 Special image database management systems and high-level picture description/manipulation machines are needed for fast image information retrieval;

5 PRIP computers need to utilize the latest hardware components and available software packages.

2.5 *Existing System Architectures*

The architectures of existing PRIP machines can be divided into three categories: SIMD array processors, pipelined vector processors, and MIMD multiprocessor systems.

Recent advances in microelectronic technology have triggered the idea of implementing some PRIP algorithms directly in hardware and some direct implementations have been built. VLSI pattern recognizers offer high speed and reliability, both of which are useful in realtime, on-line pictorial information processing. Many attempts have recently been made to develop special VLSI and VHSIC devices for signal/image processing and pattern recognition. Some of these approaches involved large-scale matrix computations and syntactic parsing operations.

2.6 *Recommendation*

The range and importance of application of pattern recognition and image processing make it clear that this area needs continuing and high-level research attention. Even the most cursory comparison of the demands for computational power with the current capability quickly shows that existing hardware is inadequate for most of the important problems.

Although it is not clear how much processing power will ultimately be enough, it is clear that this is orders of magnitude more than currently is available cost-effectively. Thus, the subgroup can recommend without reservation that a program of research be pursued which has as its objective a rapid increase in the computational power of computer hardware for pattern recognition and image processing. This program should focus not only on increased performance to satisfy the most demanding PRIP problems but also on reducing the cost of PRIP hardware for less demanding applications.

Specifically, the subgroup recommends that the following research be supported:

- Research which will qualitatively identify the computational requirements (hardware and software) of PRIP applications spanning areas of national interest,
- Research which will identify the architectural alternatives for PRIP hardware and which will provide quantitative measures for evaluating these architectures with respect to the computational requirements,
- Research which will permit systematic and rapid mapping of the architectural alternatives into VHSIC and VLSI hardware,
- Continuing research into VHSIC, VLSI and alternative device technologies which will be needed for implementing faster and/or lower cost PRIP architectures,
- Research in parallel computer architectures for PRIP applications since it is clear that parallelism is essential for meeting near-term needs with existing or foreseeable device technologies,
- Continuing research in techniques for compressing and decompressing image data for more efficient communications, storage and display,
- Research which has as its objective the development of more efficient digital representations for describing, manipulating and storing images,
- Research into computer languages for describing, manipulating and processing image data,
- Research into back-end image database machines and associated database structures, management systems, and query languages for on-line storage of massive quantities of image data,
- While representing and processing data in digital form is clearly important in the near term, abstraction is necessary to avoid such large computational and storage requirements. Research should be undertaken which explores alternatives to digital representation and processing image data.

3.0 *RESEARCH IN SUPPORT OF OFFICE AUTOMATION*

3.1 *Overview*

In this brief overview of the office automation area, we attempt to identify the key elements that comprise current and future office automation systems, outline the major trends which are becoming evident, and focus on the major inhibitors to continued growth. Specifically, we will identify those areas where economic forces are motivating adequate industrial research and those areas where they are not.

Five key elements comprise today's office automation environment:

- Workstations
- Networks that allow workstations to communicate
- Centralized services
- Applications
- Architectures which unify the office automation environment.

3.2 *Workstations*

A workstation, as presently conceived in the office automation environment, is comprised of several elements. It includes a display with associated keyboard, processor and processor-memory; a printer for hard-copy output; and some form of storage media apart from the processor memory. It may also contain a scanner for entering textual or graphic information which is not directly machine readable, and possibly a telephone. A network interface may also be provided to permit the workstations to communicate with other workstations, larger computers, or

centralized services. Each of these elements and the trends associated with their further development is discussed below.

3.2.1 *Displays*

In today's marketplace, there are three major types of displays that are competing for space on the office desk. The first type is a main-frame interactive display, for example, a VT-100 or a 3270-like display, which is connected to a main-frame application. The second type of display is typically associated with a word-processing system. The third type of display is the part of the personal computer. In the future, each of these displays will need to perform all three functions: personal computing, main-frame interaction and word-processing functions.

In today's environment, displays are typically alpha/numeric and are rapidly moving to alpha/mosiacs. This trend is driven by the videotex environment. In addition, higher function displays, able to handle limited graphics, are moving into the office environment. Finally, there are bit map displays that are able to provide high quality graphics.

The trend in the processor area is rapidly increasing performance requirements driven by the convergence of function into the display and the need for more processing power to facilitate an easier-to-use interface. In today's display/processor complex, one typically finds a microprocessor that is capable of executing between 250 and 350 thousand instructions per second. Certainly, by the 1985 time frame, this requirement for performance will have grown to a million instructions per second.

3.2.2 *Printers*

In today's office automation environment, there are three levels of printers: printers which are of draft quality, based primarily on the matrix printers of the data processing world; quality printers which use the daisy-wheel technology for letter-quality output; and bit-map printers, typically laser-based, for very high quality output. Additional technologies are being investigated for these three printer design points.

3.2.3 *Storage Media*

We have seen the rapid evolution of the diskette from capacities in the range of less than 160K bytes to a million or more bytes today. In addition, hard files, which in the past were associated with large systems, are now readily available in the 5- to 10-megabyte range. Concurrent with this increase in capacity, cost per byte of storage has reduced.

3.2.4 *Telephone*

In today's environment, the telephone is a separate device from the office workstation. However, in the marketplace there are telephones which include a small screen. At the same time, we see workstations incorporating telephone functions, including the required analog-to-digital and digital-to-analog conversions. Development is progressing in two complementary directions. One is augmenting the telephone with workstation capabilities and the other is augmenting the workstation with telephone capabilities. The telephone function itself is also evolving. Very basic functions like call-forwarding, are giving way to voice store-and-forward capabilities that elimate much of the non-productive delay encountered today in using the telephone.

3.2.5 *Scanners*

Devices are being introduced that can scan a document to produce a bit-map image that can be stored in a computer system. This allows documents that were not keyed into a system to participate in the rest of the office automation environment.

Given these major trends, what inhibitors exist for office automation workstations? From the display point of view, the inhibitor is the cost associated with the transition from low-resolution character-oriented displays to the high-resolution bit-map-oriented multifunction displays of the future. From the computer point of view, it is the apparently neverending quest for high performance microprocessors. Both of these inhibitor areas are being adequately addressed by industry, and, hence, are perhaps a lower priority for university/government funding. The same is true for the printer area, media area, phone area and, we believe, the facsimile area. Hence, we believe that for the workstation area, economic forces in industry will adequately cover the problem from a hardware point of view.

3.3 *Network Communications*

The major elements of networks that allow office workstations to communicate are the teleprocessing systems (leased or switched) and local-area networks. These are receiving wide attention in industry as well as the university/government sector. One major trend evolving in this area is the evolution of dynamic networking structures to which workstations and links can be added dynamically, and which adaptively route traffic to achieve proper load balancing. A second trend is the integration of local-area networks into the teleprocessing subsystems using an overall networking architecture which is transparent to workstations attached to the system. A third trend is the evolution of much larger networks requiring enhanced network management facilities. An example is the recently announced IBM/Carnegie-Mellon University joint development activity which plans to install 8000 personal workstations in a campus environment.

An important inhibitor to network growth is ability to manage larger networks. This inhibitor can best be studied in environments where large networks exist and can be measured. Another inhibitor is the need for protocol conversions between various network architectures. The research required to solve the internetworking problem can be done on a small scale at multiple universities. Finally, we believe that networking research should move from an environment of evaluating new protocols and topologies to an environment of understanding network management problems

3.4 *Centralized Services*

In an office automation environment, certain services must be centralized either for economic reasons or control reasons. Typical services that are centralized for economic reasons are file servers, printer servers, communication servers and perhaps audio input/output servers. Services that must be centralized for control reasons include directories, name servers, library servers, etc. The major trend in centralized services, particularly in environments involving large networks, is the ability to provide centralized services with very high availability. For example, in the file

area this might involve several file servers with redundant replicated data, which leads directly to the well-known distributed database problems.

We believe there are no major inhibitors in this area from the point of view of computer science research other than those associated with maintaining consistency and concurrency in distributed database management systems.

3.5 *Applications*

The key applications that are evolving in the office automation environment certainly start with text entry and editing. However, the text entry and editing environment is rapidly moving to include linguistic checking which ranges from spelling verification to grammar and syntax checking.

Moving from text entry and edit, the next requirement is to integrate data with text to produce the kind of mass-mail letters we all receive in the mail. This implies a capability to query a database and then to produce documents or letters that are dependent upon the result of that query.

Next in the list of key applications are those applications that produce documents involving text, graphics, and images. These may have aural annotation. Another key element is the personal computing spread-sheet application. Activity management applications are important; this includes calendaring and scheduling algorithms as well as the kind of telephone management functions discussed earlier. Finally, office applications can include forms processing and forms flow control.

Major inhibitors to the continued development and use of these applications are the lack of a consistent user interface across the range of applications and the lack of a consistent method of data exchange between applications.

3.6 *Architecture*

Four levels of architecture are required to bring together a unified office automation system. These provide for a single method of interconnecting systems, interchanging data and documents between systems, and an interface to various applications. These levels are: The transport architecture, the distribution architecture, the architecture for specifying the contents of documents, and the architecture for describing the user interface to various applications. Each of these architectural levels is being defined by industry.

A major inhibitor in this area, which is certainly appropriate for continued government/university attention, is the development and evaluation of better user interfaces, particularly as we move from a text orientation to a multimedia orientation. Another major inhibitor is the lack of standards so that in multi-vendor office automation environments, documents and data can be moved freely among systems and applications.

3.7 *Summary*

In the future, when the office automation capabilities we now envision are broadly available, new demands will arise. The professional, well served by text editing, communications and retrieval service, will begin to turn to his "desk-top world" for help in his specific task-oriented domain. These needs will again raise the challenging problems of language and process development.

It is not fruitful for the research community to think of developing "professional applications", as there are too many of them. Rather, the concern should be for developing *tools* for applications development. Both the richness and complexity of the professional's "work space" and the relative computer naivete of non-computer professionals act to make this a most difficult problem and one in which industrial firms are making only slow progress. These applications development tools are a realistic candidate for NSF support.

In summary, we have examined the key elements of office automation and have pointed out those areas where we believe industry has adequately covered the research requirements for the next 5 years. We also pointed out those areas where we believe government and university attention should focus.

4.0 *RESEARCH TO SUPPORT INCREASES IN SOFTWARE PRODUCTIVITY*

4.1 *Why Address Software Productivity?*

Pervasive to all fields of endeavor in Computer Science and Information Technology is an underlying inhibitor to progress: the lack of an adequate number of professionals. With growing demand, this supply shortage will likely get worse in the 1980's. Not only are systems development and applications development backlogs growing, but essential Information Technology R&D is being constrained. The United States, which has had an unchallenged superiority in software, may find that superiority jeopardized by these constraints.

There are two solutions to this problem. First, more people can be encouraged to enter the Computer Science profession. However, even under ideal conditions, additional professionals may not meet the demand. The second approach is to increase the productivity of the professionals in the field. This is the domain of software productivity. A trivial one-percent increase in the productivity of every existing professional would represent the equivalent of adding thousands of new professionals.

4.2 *Software Productivity and Software Engineering*

There is a strong coupling between software productivity and software engineering. The potential for productivity improvement through automated tools - particularly at the front end of the development process - is clearly significant. Formal requirements definition, formal specifications, abstract prototypes, reusable design libraries, are a few of the approaches explored in modern software engineering. However, basic questions of cost and productivity tradeoffs with these technologies are not yet well understood.

There are many cases where new programming environments and associated tools offer such dramatic improvement that evaluation is easy. However, many proposals involve tradeoffs which cannot be simply decided on *a priori* and some metrics are needed.

4.3 *Productivity and Quality*

An absolutely fundamental question is how productivity relates to quality. Some very preliminary IBM data show that post-ship quality directly correlates with pre-ship productivity. Since quality and productivity are perhaps the two basic parameters of most interest to the programming profession, we need to better understand their

interrelationship.

4.4 *Some Strategic Research Directions*

Many fruitful areas of research should be pursued. Since software development is a labor intensive effort, significant gains can come from:

1 Eliminating, through automation, the non-creative tasks of development.

2 Making creative tasks more efficient, with better tools, better methodologies, optimal machine support.

3 Decreasing the non-productive task of error correction in development and maintenance, by providing tools and methodologies for high-quality development.

4 Educating (re-educating) those professionals who have not been exposed to modern software engineering practices.

4.5 *Software Productivity is a People Problem*

The fundamental problem is to increase the productivity of people: software systems developers, software applications developers, software maintenance people, etc. A related problem is the productivity of end users, but this involves a separate set of questions on product usability, common user interfaces, common databases and standard communication protocols, which are better addressed by other direction statements.

4.6 *Dimensions of Software Productivity*

There are several views of software productivity, from small to large, each deserving of investigation. Productivity of the individual professional is highly variable and poorly understood. Next is functional productivity of design, development, test, etc. Next is total development productivity, i.e., all the work necessary to produce a product. Next is total product life-cycle productivity encompassing both development and post-development activities of installation, maintenance and update.

4.7 *Why Research is Needed on Software Productivity*

The state-of-the-art of software productivity is primitive. Professionals intuitively believe that there is a wide range of individual and group productivity; that there are key variables which influence productivity; that productivity has not significantly changed in the last decade; that there is a critical need to improve productivity; that there is potential to achieve this improvement.

However, these beliefs are subjective and difficult to substantiate. There is virtually no objective evidence backed up by comprehensive data collection, controlled experimentation, and comparative evaluation. The critically important field of software productivity needs to be pulled out of the dark ages of visceral opinion into the modern world of objective, scientific evaluation.

Research is needed to *establish* an objective discipline of software productivity, because none exists today. This is unlike most fields of Information Technology, where disciplines have been well formulated, and the theoretical foundations well established.

4.8 *First Steps in Establishing a Discipline*

What makes software productivity so difficult to analyze is the extreme number of variables which influence productivity. There are product variables (e.g., size, complexity); product environment variables (e.g., requirements stability, interface controls, testing complexity); development process variables (e.g., planning rigor, support tools, computer support); people variables (e.g., individual experience and capability, management capability), to name a very few.

Compounding this awesome multivariable problem is the disconcerting fact that there is no standard definition of software productivity. Everyone defines it differently, and measures it differently. As a result, the few measurements which do exist cannot be compared. Even such "simple" parameters as lines of code, people, months or dollars can have extremely wide variations based on what the measurer includes or excludes.

4.9 *Interdisciplinary Approach May Be Beneficial*

Since software productivity is a people problem, it may well benefit from those disciplines which specialize in individual and group behaviors: Psychologists, Social Scientists, Behavioral Scientists, etc. Computer professionals who are living with the problems of poor productivity may be too close to the trees to see the forest. Other professionals will bring objectivity, perspective and fresh insights to the problem.

4.10 *Tactical Approach to Software Productivity Research*

Pragmatically, this is a field where trying to do too much too soon would be a mistake. More beneficial than a few massive research projects would be several modest projects attacking aspects of the disciplinary and strategic areas mentioned above. This approach would provide the initial foundation upon which subsequent research could build:

It would be wise to heed Lord Kelvin's advice:

"When you can measure what you are speaking about, and express it in numbers, you know something about it; but when you cannot measure it, when you cannot express it in numbers, your knowledge is of a meager and unsatisfactory kind."

4.11 *Some Basic Questions of Software Productivity*

- How is software productivity defined? (Need more than one definition)
- Major variables which influence software productivity?
- Quantification of software productivity? (across several dimensions)
- Software productivity improvement objective for 1990?
- Why such variability in individual productivity?
- How best to eliminate non-creative tasks?
- Best tools to assist creative tasks?
- Optimal developer/support level?
- Relationship of quality to productivity?

5.0 *RESEARCH TO SUPPORT THE DESIGN OF LARGE-SCALE SYSTEMS EMPLOYING COMPUTERS*

An urgent national need exists for the engineering design and utilization of large-scale systems which employ

substantial numbers of computers to perform sophisticated functions. One may conveniently associate three categories with this need:

1 The development of computer systems and interconnecting networks for use in engineering design;

2 Real-time processes under computer system control in an environment such as the totally automated factory; and

3 The real-time iterative interaction between the design process of (A) and the plant operation of (B) to achieve feedback for optimization and/or error correction. Some of the dimensions of the research effort envisioned in this area, as well as an appreciation of its need, can be obtained by some specific examples of topics which need attack. Manufacturing processes and complete manufacturing plants obviously involve very-large-scale systems whose design and utilization can be substantially improved by suitable knowledge developed in the research area. Automated production of integrated electronic systems and mechanical systems, including vehicles, are familiar examples. Perhaps a less obvious need is use of large-scale computer systems for prediction of environmental impact of measures taken to manage water resources (floods, water quality, reservoir operation, and toxic contamination of ground water) and environmental quality (noise in urban areas, air and water pollution). These tasks frequently include map digitization, registration, overlaying and coupling simulation and optimization models to map databases. Further applications occur in transportation-systems analysis dealing with questions of the least costly or least hazardous routing of cargo, design of transportation networks and simulation of urban goods movement. Numerous additional examples from diverse fields of national interest can also be identified.

In dealing with the problem areas of interest, there is need for the availability of relevant models, software, and data expressed in a form aimed at assuring portability. These are needed in addition to protocols for information exchange among interested users. Other research items of interest include aspects of software engineering design and use permitting reuse or multiple use of sophisticated software packages; the methodologies of model construction from the scale of electronic devices to automated plants; the design of databases having wide applicability; problems associated with system interfaces; problems of real-time feedback in a complete computer-controlled plant, including feedback to the design process; and fault-tolerant operation, and the human operator-system interface in systems with a large number of computers. Each of these items has been investigated already to some degree, but not to the extent judged to be necessary to meet the need which has been expressed. For example, some of the modeling techniques which have evolved to support the design of complex computer programs and complex computer systems may be applicable to general engineering design. It appears, however, that there is at least an order of magnitude of greater complexity required to create models of manufacturing systems, for example. The traditional mode of NSF research support, particularly for work in colleges and universities, is characterized by support of work in pure and applied science performed by individual scientists and engineers, working alone or in small teams in an academic environment. Work related to the problems cited above which has received NSF support has been limited generally to advancing scientific knowledge with regard to individual *elements* of large-scale systems, such as robotics, computer vision, and parallel processors. It seems evident that, for the nation to mount a fruitful program of research in the area of interest utilizing fully the talents of academic investigators, a research function beyond what is currently available needs to be supported. This is necessary in view of the anticipated need for an interdisciplinary and interinstitutional cooperative effort of substantial size required to deal successfully with the number of disciplines and the magnitude of the problems involved.

6.0 *MANPOWER/RESOURCE REQUIREMENTS*

Near the conclusion of its work, the group gave consideration to manpower requirements and resource requirements related to the problem areas under discussion.

The first issue which arose was the cultural shock created by unprecedented rates of technological change. In particular, this has led to deep shortages of technically competent computer-literate engineers, scientists and managers. Industry needs served to deplete the supply of graduate students continuing on to Ph.D.'s and to careers in academia. This was further aggravated by the poor state of equipment and facilities in universities relative to industry. Some corrective measures are now being taken, but there is little confidence that a good solution path has been identified. There are a number of issues which need careful attention and innovation:

- There is a widespread need for retraining of personnel in all walks of life which are impacted by computers. This is a massive task and is necessary both to increase the level of competence and to prevent demoralization of the work force;
- There is a need for continuing education which makes use of available technology to a much greater extent. Teleconferencing, closed-circuit TV courses, videocassette-based instruction, Plato-like tutoring systems and the use of adjunct faculty from nearby industry were suggested;
- Special attention should be given to providing good educational software for the personal computers which are reaching homes and schools through the consumer market,
- There were several comments about the sad state of our elementary and secondary education systems and the importance of doing something, but there was no way for us to grapple with that issue except to encourage local attention to it;
- There was exposed a particular need to provide early warning of oncoming technological changes to groups which would be most impacted by them.

7.0 *INTERFACE INSTITUTIONAL REQUIREMENTS*

Consideration was given to what institutions like NSF or DOD might do to alleviate some of the problems which have been exposed, given that budgetary limitations will not allow provisions of resources to every group.

The primary suggestion was to create more sharable

resources, e.g., to establish shared centers for testbed research on automated manufacturing or super computing reachable by investigators. In the latter case, work can be accomplished remotely through computer communications. In the former case, national accelerator or radio-astronomy centers offer models. It is possible that there could be an impact made in a combination of available infrastructures at the national laboratories and volunteer universities; expanded communication networks including satellites; and a cadre of people interested in serving computer-aided engineering and computer-aided science by planning and facilitating the sharing of large facilities. The success of the ARPA Network offers a precedent for such successful programs.

A second suggestion is that NSF put additional resources into support of research affecting the user-interface to computer systems, i.e., graphics, help systems, speech input and output, common languages, etc. Obviously good research in this area serves to positively impact all areas of use.

8.0 *SUMMARY PARAGRAPHS*

8.1 *Office Automation*

The substantial advances in integrated electronics, distributed computing and systems understanding have enabled the now explosive growth of office automation. Because of its great commercial value, the field has been embraced and is heavily supported by the private sector (with aggressive investments by companies large and small). Accordingly, in a time of economic stringency, it is our view that NSF support should be focused most sharply on the vexing, fundamental problems that will remain even after the basic hardware and software support systems are in place. These problems, some of the most enduring in computer science, have to do with the central issues of language and process development. We still have few effective ways for computer-naive professionals to express their domain-specific needs to the powerful engines sitting on the desk in front of them. Applications languages and development tools remain a major research item.

8.2 *Pattern Recognition and Image Processing*

There is an important need to increase the speed and reduce the cost of hardware used for pattern recognition and image processing. Research should continue in highly parallel architectures for achieving higher speeds. Research which will permit systematic and rapid mapping of architectural alternatives into VHSIC and VLSI hardware is also needed. Research should also be conducted which addresses the storage and retrieval of huge numbers of images in back-end databases. More efficient digital representations and languages for manipulating them are also needed. Research should also be undertaken which explores alternatives to digital representation and processing of images.

8.3 *Software Productivity*

There is a great need for software productivity improvement, pervasive to all areas of Information Technology. There is also great potential for improvement, specifically from software engineering tools and methodologies. However, there is a lack of coupling between software engineering research and the exploitation of that research in software development environments. Modest research projects, not requiring massive support resources necessary for other areas of research, could focus on this coupling and result in significant productivity benefits.

8.4 *Large-Scale Systems*

We have concluded that it is very much in the national interest to develop the knowledge-base necessary to design large-scale systems which take full advantage of the capabilities of modern computer technology. To accomplish this objective, provision beyond support for individual investigators must be made for:

1. construction, support, and coordination of selected large-scale systems used as testbeds for research and shared by researchers;

2. coordination of funding by government and the private sector (including grants of equipment) in support of this research area;

3. aiding the achievement of overall research objectives by serving as a catalyst to the combined efforts of many investigators. The NSF is an appropriate body to serve as the lead organization in assuring that these functions are addressed.

We strongly recommend that a program of research responsive to this need be initiated at the earliest practicable date.

8.5 *Peer Review and Program Review Processes*

In the course of its deliberations, the group noted examples of substantial differences in significance and sophistication of research conducted in universities and industry. This is not to imply, however, that either one or the other can be expected to excel consistently in quality of research results. Rather, the significant factor in the difference in results is the economic forces that drive the effort that can be supported in certain industrial research programs. Office automation, for example, is an area where industry research is currently the dominant force.

The group concluded from its deliberations that minor modification in the operating practices of the NSF could contribute significantly to the effectiveness of its total program of research support in computer science, computer engineering and information science. The change is to involve representation from appropriate areas of industry to a substantial degree in peer-review and program-review committees associated with those program areas in which significant industrial research is being conducted.

ROBOTICS

An Assessment of the State-of-the-Art and
Recommendations for Future Directions

Prepared* for
NSF Information Technology Workshop
Xerox International Center
Leesburg, Virginia

January 5-7, 1983

by

Jacob T. Schwartz, Chairman	New York University
Herbert Freeman	Rensselaer Polytechnic Institute
John E. Hopcroft	Cornell University
Stuart Miller	General Electric Company
Peter M. Will	Schlumberger Well Services

*Adapted from Jacob T. Schwartz and David Grossman, "The Next Generation of Robots," in *Outlook for Science and Technology: The Next Five Years.* Vol. III. Washington, D.C.: National Academy of Sciences, 1982, pp. 167-190.

ROBOTICS

Table of Contents

1.0 *INTRODUCTION*

Robotics is the field concerned with the design and use of "intelligent" machines. As such it overlaps many of the established disciplines -- control theory, artificial intelligence, mechanical engineering, computer science and engineering, manufacturing technology, physics, material science, and industrial engineering. Its immediate prospects are bound up with the rapidly growing industrial efforts to improve productivity by extending and "roboticizing" the present techniques of automation.

For the purposes of this chapter, robots can be defined as computer-controlled devices that reproduce human sensory, manipulative, and self-transport abilities well enough to perform useful work. (Of course, robot sensors sometimes surpass human capabilities in certain applications, and they may work in very different ways.) Proceeding by anthropomorphic analogy, the components of these machines can be grouped into six main categories:

Arms--robot manipulators

Legs--robot vehicles

Eyes--robot vision systems

Ears--computerized speech recognition systems

Touch--tactile sensors and artificial "skins"

Smell--smoke detectors and chemical sensors of other kinds.

There are also other, less human kinds of sensors. Commercial robot systems are available in all of these categories, and systems that integrate multiple capabilities are beginning to appear on the market.

In addition, there are many kinds of nonrobot systems that handle tasks requiring some degree of intelligence, for example, "expert systems" of various kinds. These systems were discussed in the second *Outlook for Science and Technology--The Next Five Years* (W.H. Freeman and Company, 1981, pp. 747-753.)

In examining the problems that are currently faced by the robot designer and robot user, it appears that some of the most central and most difficult ones are those involving languages for specifying robot tasks, planning of actions and motions, and creating geometric models of complex objects and environments. These problem areas are those for which computer science and engineering appear best prepared to make the needed contributions.

Although it can be expected to draw upon many other branches of computer science, robotics will have a different flavor from any of them because, in robotics, computer science must go beyond the combinatorial and symbolic manipulations that have been its principal concerns until now. To address the problems of robotics, computer scientists will have to confront the geometric, dynamic, and physical realities of three-dimensional space. This confrontation can be expected to generate a great deal of new science, and robotics should be as central to the next few decades of research in computer science as language, compiler, and system-related investigations have been to the past two decades.

Robotics is exciting precisely because of the broad mix of computer science and traditional science that will be involved in it. Through it, computer science will learn to deal with the modeling of three-dimensional objects, with partly physical questions such as the motion of bodies in situations affected by friction and with many engineering issues. System integration problems of great challenge will be encountered and will tax all the resources of the programming language designer and software engineer. Finally, robotics will involve computer science with expensive and dirty experiments of a type familiar to enginering and experimental physics groups, but not previously attempted by computer science departments. Judicious choice of such experiments will challenge computer science researchers, and how to approve and pay the bill for these experiments will be a problem for funding agencies.

2.0 *HARDWARE AND SOFTWARE LIMITATIONS*

OF CURRENT ROBOT SYSTEMS

Both the sensory capabilities of robots and their ability to deal with unexpected events are as yet quite limited. For this reason, robots presently are effective only in highly structured environments in which the position and path of motion of all objects are known fairly precisely at all times. At present, this structural constraint largely confines robots to industrial rather than more arbitrary environments, and mostly to work with hard materials (metal, wood, hard plastics), rather than with soft, flexible objects (cloth, vinyl) that are hard to control. It is the gradual removal of this constraint that, broadly speaking, drives computer science research in robotics. Besides its obvious economic significances, this corresponds well with the fundamental trend in all programming activities.

In spite of these limitations, robotics is expanding. This is partly the result of its mastery of such commercially significant activities as spot welding and spray painting and an accumulation of technical expertise over the past decade. It is also a consequence of the microelectronics revolution which already has reduced the cost of computer control drastically, and which, in years to come, should produce many integrated "miracle chips," embodying advanced sensory and end-effector control functions. The combinat' of electronics, technology, and software science represented by robotics can be expected to lead industry into an age of electronic factories with drastically diminished manufacturing labor forces. However, all of this lies some decades in the future. Even the use of robots to perform the majority of industrial operations is decades away. For the near term, the potential applications of robots are extensive, but not unlimited.

3.0 *APPLICATIONS: ECONOMIC CONSIDERATIONS*

The automated industrial environment is one in which workpieces, that is, items being manufactured, move along controlled paths through a variety of steps, such as crimping, shaping, welding, melting, cutting, stamping, spraying, painting, and assembling. As workpieces move along an automated manufacturing line, they are placed accurately on pallets or pushed into positions between fixed walls, pipes, and chutes, as well as rods, plates, cams, and other moving members. This requires detailed sculpturing of the geometric environment in which the workpieces move. The specialized fittings and mechanical tooling demanded by this process make fixed factory automation very expensive and ill-equipped to cope with changing product lines. Robotic technology aims to use a few standardized but adaptive mechanisms, of which the robot manipulator is the prototype, to replace these expensive special purpose fittings. Of course, robot manipulators will continue to use many of the tools presently employed in automated manufacture.

4.0 *ROBOT MANIPULATORS*

Generally speaking, robot manipulators (for example, industrial screwdrivers) perform with high efficiency special motions that grasp or otherwise acquire objects (such as electromagnets and vacuum lifts) or that apply special physical processes to workpieces (for example, spray gun, welding gun, industrial shears, stamping press, drill press.)

Fixed automation can be expected to remain more advantageous for high-volume manufacture than programmable robot equipment. This is because specially designed mechanical equipment can move more rapidly than general-purpose robot manipulators, and it is, therefore, more productive once the high initial cost of setting up an automated production line has been amortized. On the other hand, for manufacturing individual items or very small batches, manual production often will be less expensive than a robot control program. Thus, the area most favorable to robotics is medium-quantity batch manufacture; in other words, the manufacture of items in batches numbering several hundred to several thousand. In this connection, it is quite important that robot systems be flexible and easy to adapt to new applications. The more the setup costs associated with typical applications can be reduced, the smaller will be the size of the least robot manufacturing run that remains economical.

The notion of flexibility has two aspects:

A. Robot mechanisms must be universal, e.g., manipulators have six or more degrees of freedom to permit arbitrary positions/orientations to be attained, even in the presence of fixed obstacles;

B. The programming system which drives a robot must be structured to allow several different classes of users to use it as conveniently as possible. The system must accommodate the requirements of

Robot techniques also will be advantageous when larger numbers of similar but not identical items need to be manufactured, provided that the variations between items are not too large to be accommodated comfortably without very sophisticated computer control. The furniture manufacturing industry illustrates this situation: many items of furniture, especially fine furniture, are produced in relatively small batches as orders for specific designs are received and as shipments of wood with matching grains come in.

Today's robot manipulators are usually sedentary, stand-alone arms with limited precision and very little sensory capability. Many improvements in these relatively primitive devices are possible but, even in their present state, they support applications with growing economic values. The most important current industrial applications of robot equipment are handling materials, loading/unloading machines, transport (surrogate conveyor), palletizing/depalletizing/kit packing, processing and fabrication, welding (spot and arc), spray painting, drilling, assembly (parts mating), and testing (dimensional, continuity).

Table 1 describes the ways in which one of the simpler classes of robot devices is used. This class consists of the "playback robots," which simply repeat a sequence of motions through which they have been led. (These simple robots can be seen in major automobile plants.)

TABLE 1
Breakdown of shipments of Playback robots by work process, March-September, 1979

Work Process	*No. of units*	*Percentage of value*
Spot welding	57.1%	45.1%
Arc welding	18.8	26.0
Spray painting	11.3	17.8
Other	12.8	11.1

Source: Paul Aron. "Robotics in Japan," Paul Aron Reports, No. 22 (July 3, 1980). Published by Daiwa Securities America, Inc., New York.

5.0 *ROBOTS IN THE UNITED STATES AND JAPAN*

Although France, West Germany, and Sweden all have active robotic research groups and are introducing robots rapidly into industrial settings, robot use is most advanced in Japan. It is sobering to reflect that, even though much more than half of all robotics research and development before 1975 was performed in the United States, Japan now leads the United States in applications. Among other things, this illustrates the fundamental importance of technology transfer mechanisms deeply rooted in a nation's educational and economic systems. Table 2 shows how far behind Japan the United States has fallen in the wider user of industrial robots.

TABLE 2
Comparison of industrial robots in the U.S. and Japan, 1980 (using U.S. definition)*

	Japan	*U.S.*
Production in units	3,200	1,269
Production in value ($ million)	180	100
Installed operating units	11,250	4,370

*Japan recognizes six classes of robots, while the U.S. recognizes four. The U.S. definition covers variable-sequence robots, playback robots, numerically controlled robots and independent robots. Japan adds manual-manipulator and fixed-sequence robots.

Source: Paul Aron. "Robots Revisited: One Year Later," Paul Aron Reports, No. 25(July 21, 1981). Published by Daiwa Securities America, Inc., New York.

Most (nearly 70 percent) of the installed Japanese robot manipulators are of the very rudimentary "fixed sequence" class; in other words, they execute repetitively a sequence of motions that is not easily changed. "Playback" robots differ from these in that they can be "taught" new motions by being led manually through the sequence of points to be traversed. Playback robots constitute roughly 8 percent of installed Japanese robots, but 16 percent (by dollar value) of the Japanese robots shipped in 1979. More sophisticated robots of various kinds constitute roughly 12 percent of those shipped in 1979.

Playback robots are essentially clockwork mechanisms in which cams have been replaced by microcode. They maintain no internal model of the objects being manipulated or of the results of the manipulations that they carry out. Hence they are unable to generate any error recovery sequence when a manipulation fails, e.g., when an object is not found in its expected position. Because of this rigidity, playback robots are only useful in highly structured, hence expensive environments.

As background to these figures, it should be noted that manufacturing accounts for roughly 21 percent, and durable goods manufacture for roughly 13 percent, of the U.S. gross national product.

6.0 *LIMITS ON TODAY'S ROBOTS*

As already noted, currently available robots are quite limited both in their sensory capabilities and in their ability to deal with unforeseen contingencies. A corollary is that, as items to be processed enter the robot workspace, either they must be in their proper order and orientation as a result of preceding operations, or order must be imposed by passing them through parts feeders or chutes of an appropriate geometry. Thus, as these workpieces move in a roboticized factory, either control over their position must be maintained, or reorientation operations must be performed repeatedly. A central aim of industrial robotic research is to relax these constraints--that is, to find ways of dealing with less and less structured environments. Accomplishing this will require development of better sensors and improvement of procedures for analyzing sensor-generated data, object recognition algorithms, environment-modeling software, and computerized replanning techniques. The "bin picking" problem, the problem of locating a part in a disordered bin of parts so that it can be lifted out and given a prespecified orientation, is typical of the problems that robot designers face in dealing with disorder.

7.0 *CHARACTERISTICS OF MANIPULATORS*

Current, general-purpose robot manipulators have one arm with three to six independently controllable "joints" (involving motor, encoder, and controller) plus a gripper that can be opened or closed. Dozens of manufacturers are offering manipulators of this kind. Generally, these arms are not mobile, but a few experimental mobile robots have been built. However, today's robot carts are generally armless, stand-alone vehicles that follow preset paths. Typical uses are delivering office mail or hospital meals and linen. The six degrees of freedom that are typical of current robot manipulators suffice to position the robot's gripper, or a tool or workpiece that it is holding, at any arbitrary point within the robot's accessible space, with an arbitrary rotational orientation. Robots with a large number of degrees of freedom (e.g., elephant's trunk manipulators) have been conceived, but these pose deep problems of kinematic and dynamical control, even to the extent of making contact with deep mathematical problems of 'catastrophe' theory (classification theory of singularities of smooth mappings.) Though this subject area raises a wealth of research issues, only one or two papers have as yet addressed it. The gripping hand mounted at the end of a manipulator arm is often equipped with a few simple sensors, for example, strain gauges that sense contact with external bodies and grip forces, or photocells that detect the presence of an object between the gripper fingers. In some setups, one or more television cameras are mounted in positions that allow them to observe the robot arm or hand and the objects that the manipulator

approaches; information derived from analysis of the images is passed to the program controlling the robot. Control is ordinarily exercised by an inexpensive mini- or microcomputer.

The "reach" of a typical robot manipulator may vary from 1 to as many as 10 feet; the "payload" that it can lift ranges from a few ounces to several thousand pounds.

Manipulators vary in price from a few thousand dollars for small, simple, slow-moving arms without sensory capabilities, intended for light educational use, to roughly a hundred thousand dollars for fast, sensor-equipped industrial systems capable of highly precise mechanical movements (for example, returning to within 0.01 of an inch of a prespecified position.) The working volume of a large industrial manipulator, in other words, the volume of space it can reach, may be as large as a cube 10 feet on a side. Speeds as high as 5 feet per second are not unusual.

Although the sophistication of robot control systems can be expected to increase rapidly because of the growing availability of powerful microprocessors, much of the industrial robot equipment in use today cannot respond flexibly to changing external situations. Manipulators of the "playback" type, which simply store and repeat some specified sequence of motions, represent the extreme. More flexible systems include a mini- or microcomputer for control. In such systems, control programs written in a reasonably powerful robot programming language are stored in the computer's memory and determine the manipulator's sequence of motions. Special statements in the robot language allow information to be read from sensors and responses conditional on this sensed information to be programmed. However, even in using a programmable robot, it is often convenient to describe both the main outlines of the sequence of motions that it is to perform and the key points that these motions are to traverse, by moving the arm through these desired positions manually. For this reason, robot manipulators are generally equipped with a "teach box" with keys that allow the arm to be moved or rotated by hand. The robot's control computer then acquires the manually guided motions and integrates them with a more comprehensive program that also involves the use of sensors and sophisticated conditional responses.

As long as a manipulator arm can be moved through free space up to the precise point at which it will make contact with an external body, determination of the manner in which it is to move is relatively straightforward. Although challenging problems of dynamic control do arise when the manipulator is to be moved rapidly, especially if it is simultaneously grasping an object of appreciable mass or if it must operate in a moving coordinate frame, generally, control of unimpeded motions is not difficult. However, it is impossible to maintain absolutely precise information concerning the position of a manipulator and its workpieces at all times, if only because manipulator arms themselves will deform slightly as they move, or become worn or slightly ill-adjusted. Therefore, one needs to deal with the phenomena that arise when an object firmly gripped by a manipulator comes into contact with an object fixed in some other coordinate frame. Here, purely geometric control becomes infeasible since, if it touches an object, a manipulator moving along a rigidly prescribed path will break either the object or itself. Consider the problem of inserting a peg into a hole; if the motion of the peg is perfectly independent of the forces exerted on it by the walls of the hole, any geometric imprecision will jam the peg either at the mouth of the hole or against one of the walls.

Thus, as bodies come into contact, robotics leaves the purely geometric domain and must deal with problems of force-sensing and with hybrid motions guided, in part, geometrically, but also by compliance with external forces. An important step toward more flexible robot systems would be for the control software furnished with manipulators to provide "force-controlled" motion commands; using these, one could, for example, easily cause a manipulator to paint a stripe on the exterior surface of an automobile or apply adhesive to an aircraft window frame. However, although techniques for doing this have been investigated theoretically and demonstrated in research laboratories, motion control at this level of sophistication is not yet available as a standard feature of commercial robot equipment.

8.0 *ROBOT PROGRAMMING LANGUAGES*

If robots are to be applied widely in industry, they must be easy to use. Potential users forced to deal with rigid, hard-to-understand robot control languages will be discouraged from applying the new technology. For this reason, more powerful and user-friendly robot languages are essential.

Although we are far from knowing how to describe those built-in operations that would be most desirable for such a language, the following discussion of significant operations should help to summarize the present state of these languages and to anticipate some of the things that future robot languages are likely to provide.

8.1 *Available Languages*

The relatively undeveloped state of robotics is illustrated vividly by the fact that current robotics languages support only a handful of the primitive operations that are desirable. (In this context, "primitive" refers to built-in, efficiently implemented operations which can be initiated by a single statement of the language.) The primitive facilities that now exist in at least one commercially available robot language are roughly the following.

1 Manipulator motions can be described and controlled in XYZ axes fixed in space, or in frames defined relative to an object grasped by the manipulator. Motions passing through a known sequence of positions/orientations, with known speed, can be specified, typically by commands having the form.

MOVE ARM THRU POINTS P1,.....,Pn AT SPEED S.

2 Motions can be controlled and manipulator positions determined quite precisely. The manipulator can be put into a manually guided mode and moved to any desired position. This position can then be measured by the controlling computer, stored, and the manipulator returned to automatic mode. (As already noted, this "teach" mode or "guide-through-the-motions" approach programming ordinary robot applications.)

3 Several geometric and symbolic computations (for example, transportation of geometric data from one Cartesian frame to another, and generation of code from manually "taught" motions) are supported by the more advanced commercial robotics languages.

Using these facilities, one can, for example, teach a robot the position of various points on an automobile body by guiding the manipulator manually to these points, and then easily cause the robot to move to these same body points, even if the body, moving down an assembly line, is presented to the manipulator at a different angle. A typical command reflecting this general capability might have as its form:

MOVE ARM 3 INCHES FORWARD IN FRAME OF TOOL.

4 Tactile and visually sensed events can be detected. These can be used either to trigger interrupt routines or to terminate motions. User-defined events generated by nonstandard sensors are provided for also; for example, a signal from one or more photocells mounted near a manipulator can halt or redirect the manipulator's activity immediately (as safety concerns might require.) A typical command used for this purpose might read: WHENEVER SWITCH-1CLOSED IS DETECTED, EXECUTE SAFETY PROCEDURE.

5 Parallel processes (capable of controlling the activity of multiple arms or other effectuators) can be defined, activated, suspended, synchronized, etc.

6 Some image-analysis software is available. This is beginning to be integrated with manipulator control software, facilitating the construction of combined hand-eye systems.

The last item brings us to the forefront of robot technology now available commercially.

8.2 *Desirable Extensions to Robot Programming Languages*

Traditionally, computers deal with data objects which are either numbers (real or integer) or data structures such as arrays, lists, or trees. Robot programming languages will have to deal with a wider variety of entities, whose properties reflect much more of the geometry and physics of real world objects. It remains to uncover helpful sets of operations on these objects, and any helpful 'algebra' of such objects that may exist.

The addition of new statements to robot programming languages can be expected to reflect advances in robotic technology rather faithfully. The following are some of the additional control language facilities that would be most desirable:

1 Statements that cause a manipulator arm to move between specified positions while automatically avoiding certain obstacles or undesirable arm configurations. For arms with more than the standard six degrees of freedom, this may involve the capacity to "reach around" objects in a manner impossible for a standard arm. A robot control language designed for these purposes also should be capable of dealing correctly with situations in which the arm is gripping some object that will move with the arm. For example, one would like to be able to use a command like:

MOVE TOOL FROM FRONT-OF-BOX TO BACK-OF-BOX

leaving it to the robot to figure out the path necessary to avoid bumping into any obstacle.

2 Motion control statements specifying specialized "wobbling" or "twisting" motions useful for overcoming friction. For example:

PUSH PEG-1 1 INCH FORWARD WHILE TWISTING

3 Statements for managing force-controlled motions during which some of a manipulator's geometric coordinates move along specified paths at specified rates while other degrees of freedom adjust to the environment via sensed forces. Such statements might specify motion in contact with surfaces, edges, seams, and so on. An example would be:

SLIDE FINGER ALONG SEAM MAINTAINING 1 POUND VERTICAL FORCE.

4 Statements for guiding a self-locomoting robot.

5 Statements assisting in grip management, i.e., in determining the minimum gripping force that must be exerted in order to prevent a body of a given weight, gripped at known points, from slipping. Also necessary is some way of automatically causing the grip on an object to tighten when the object begins to slip.

6 Statements for managing a deformable or multi-fingered hand that is to grasp an object "geometrically" rather than by friction, enabling it to "surround" the body so that it cannot fall or otherwise escape without passing through part of the hand. An example might be:

GRASP PASSING MIDDLE FINGER UNDER ROD.

7 Control of groups of robots. Future automated factories will involve, not single isolated robots, but large cooperating groups of robots, associated with fixed automation devices and numerically controlled machine tools of a more traditional sort. Management of such systems raises very challenging problems of systems integration and algorithm design. If, for example, collisions are to be prevented, fast algorithms capable of analyzing the position of large numbers of moving bodies and determining whether or not any two of them overlap. If rapid motions are to be allowed, one may require more sophisticated algorithms which can very rapidly determine minimum distances between multiple moving bodies, compare them to their motion speeds, and determine whether to stop before collision remains possible. These two questions typify the many challenging problems in computational geometry which studies of robotics are bound to suggest. Moreover, even when algorithms of this type are available, their integration into major software systems able to control arbitrary numbers of arms, with associated visual and tactile sensors, robot carts, machine tools, etc., raises stupendous software-building problems, which will have to be addressed both by design of much improved languages, and by major experiments.

8 Statements for coordinating the frames of general objects moving relative to each other, including

objects whose position can be sensed but not affected by a robot's control computer. These would include statements for simultaneously monitoring the state of many sensors, for coordinating the activity of multiple robots, and for assuring that, when multiple robots enter each other's immediate vicinity, collisions are avoided. An example might be:

MOVE HAND-1 TOWARD FRONT-EDGE-OF-BALL, APPROACHING AT RATE 1 FOOT-PER-SECOND

when the "ball" in question is being held by another robot hand.

9 Statements for dynamic control of a manipulator, e.g., for causing a manipulator to strike a calibrated blow.

10 Statements for managing the activity of a small robot attached to or held by a large robot.

11 Facilities for modeling relations of attachment and of gravitational support.

12 To the above specific comments, it is worth adding the general observation that to facilitate interactive debugging it may be advantageous for robot languages to be partly interpreted rather than fully compiled. (However, run-time efficiency makes partial compilation desirable.) Many influences will tend to make robot program debugging particularly difficult. The danger of physical damage when executing undebugged programs is one of these, another is the fact that state restoration for systematic exercise of multiple branching paths through such programs raise complex issues. Though extensions of normal optimising technology may contribute substantially to the performance of robot programs, this issue has not yet begun to be examined by industrial or academic organizations.

9.0 *ROBOT SENSORS*

A variety of attached sensor devices enables robots to respond to their environment. These include ultrasonic range sensors, tactile sensors of various kinds, and visual sensors. The following discussion will concentrate on visual sensors, which are particularly important because of the high speed and nonintrusive nature of vision. However, tactile sensing can be more important than vision in some applications, for example, in the assembly of tightly fitting mechanical parts. The elastic and frictional forces critical in such applications vary so rapidly with changes in relative part position that little understanding of what is happening when a part jams during assembly can be gained by visual inspection. Instead, one needs to use strain gauges or other tactile elements to measure and respond directly to the forces that develop during assembly. This explains the importance of such devices as the Draper Laboratory's Remote Center Compliance (described below) and the strain-gauge instrumented version of it now being developed.

9.1 *Structure and Characteristics of a Vision System*

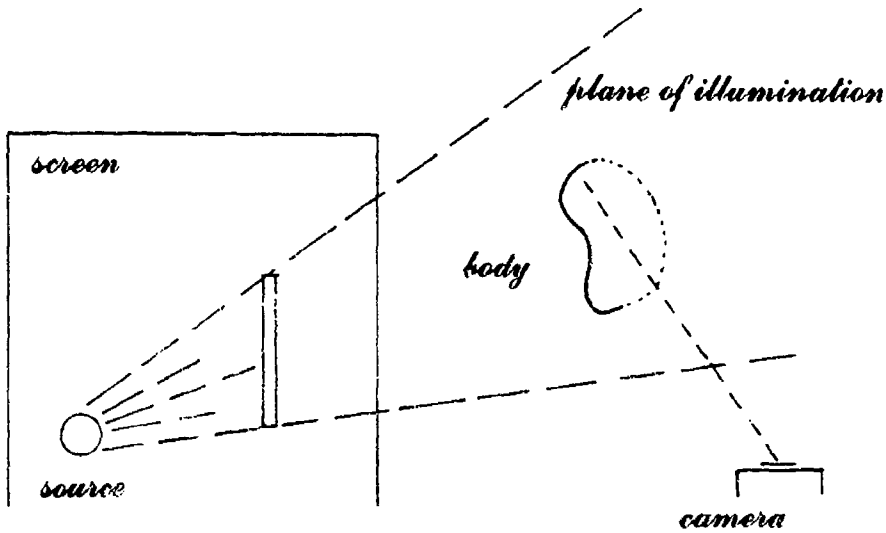
Image acquisition is, of course, a necessary prerequisite to all else. Two approaches are used currently. These are uniform illumination (like that most comfortable for the human viewer) and "structured light." At present, uniform illumination schemes, which exploit readily available television monitors to form digitized scene images, are the most common. The capabilities of such a vision system will, of course, be constrained by the quality of the images with which it works and the rate at which basic image-processing operations can be applied. The spatial resolution of the solid-state video sensors used in uniform-illumination robot vision systems of this kind is increasing rapidly. Inexpensive devices for acquiring two-dimensional arrays of 256 x 256 pixels (elementary "dots" in a picture) in less than one millisecond, and arrays of 512 x 512 pixels or finer at video rates, are available commercially, as are somewhat more expensive devices for acquiring images with up to 2,000 x 1,000 pixels.

Because of the human inability to detect video display flicker above 60 Hertz, most video systems are designed to process a single image in about 25 milliseconds. High-performance "video stream processors" are designed to perform single and binary frame operations at that same rate.

Structured-light vision schemes illuminate a scene by one or more narrow planes of light, each of which lights up some narrow, broken curve, C, on the bodies that make up the scene (see Figure 1). Since the plane of illumination is controlled, and since a line of sight drawn in a known direction from the viewing camera will intersect this plane at just one point, the position in three-dimensional space of each point on the curve C can be calculated.

By sweeping the illuminated plane at a controlled rate over a scene, one can determine the full three-dimensional location of each body surface in the scene. Even though measurement imprecisions, unwanted specular reflections, and problems of camera calibration all complicate the mathematical simplicity of this scheme, the structured-light approach, which is currently used in a few industrial locations and is being refined by researchers at Stanford Research Institute, the National Bureau of Standards, and elsewhere, seems promising.

Once an image (either a two-dimensional image or the kind of three-dimensional image of a surface that a structured-light scheme is capable of producing) has been acquired, one needs to use it to locate and recognize the bodies present in the acquired scene. The complexity of this "scene analysis" depends on the assumptions one can make concerning the scene to be analyzed. If the scene is known to contain only one object and if the object is being viewed from one of a known finite number of positions, relatively simple processing will locate and identify the body. The technique typically used to handle simple cases is to divide the digitized picture into separate "blobs" (each defined as a connected region of some characteristic threshold intensity) and to extract global features of these blobs, for example, their area, centroid, principal axes, and number and size of contained "holes." In the simplest cases, this information, which can be extracted using relatively easy bit-parallel arithmetic and Boolean operations, suffices to distinguish bodies and determine the angle from which they are being viewed.



Intersection of plane of illumination with line of observation

'3-D' or 'Striped Light' Vision

Figure 1. Structured-light approach to imaging.

In less simple cases, and especially if some of the bodies being viewed are partly obscured by other bodies in the same scene, more difficult analyses, which research has not yet fully reduced to practice, must be attempted. Basically, one tries to extract local body features (such as a person's nose seen in profile) by which an object can be identified even when most of it is not visible. A typical first step is to find such characteristic features as "edges" and "corners"; edges are defined as curves along which intensity changes rapidly, and corners as points at which the direction of an edge changes rapidly or at which several edges meet. Unambiguous detection of these image features in the presence of digital noise, breaks in perceived edges, and shadows and random bright spots is not easy, but some success has been attained by applying appropriate differencing and other picture-enhancement operations across the individual pixels of an image.

Although some of these methods have had partial success, it must be admitted that current computer vision software is of limited reliability and quite expensive computationally. Much faster and more stable picture-processing algorithms and devices are needed. Techniques for the identification of partially obscured bodies are particularly important. In order to deal with moving bodies (for example, to sense their rate of motion), higher picture-processing speeds are needed. Vision systems that can be reprogrammed easily for a wide rage of applications are also desirable but, at present, it is not clear how they can be created. It is hoped that understanding can be reached concerning the algorithms most appropriate for detecting image features and how to use them to identify objects present in a scene. This will encourage the development of dense, complex, special purpose integrated circuit (VLSI) chips, which could then perform the required optical sensing operations and geometric computations very rapidly.

Methods are needed for high-level, hierarchical modeling of three-dimensional shape. The methods should permit describing an object in progressively finer detail, as needed to identify it and distinguish it from other objects. The recently developed octree method, in which an object is modeled by iteratively subdividing a cube and "discarding" at each level those subcubes that are either wholly inside or wholly outside the object, is a step in this direction. However, there is a need for such modeling schemes that utilize iteratively natural shape features, such as describing a coffee cup as a thin-walled cylinder, closed at one end, with a ratio of length to diameter of about 3:2, and then in hierarchical fashion proceeding to progressively finer shape features. Such a scheme would make it possible to identify a new type of coffee cup, as well as to describe those features by which it differs from a cup of another type.

It appears likely that a computer vision system cannot be effective in a general environment unless it has both global and local vision. That is, it must be able to perceive the whole scene as well as any desired subsection of it. This "local" image may be obtained by zooming in on a portion of the scene or using a separate camera mounted on the manipulator arm. In such a dual-vision system it becomes information extracted from the local image to that obtained from the global image. Very little research has

thus far been done in such integrated multi-image vision systems. (The integration of tactile information into such a vision system should also be considered.)

9.2 *Applications of Computerized Image Analysis*

9.2.1 *Current Applications*

Although the capabilities of robot vision systems still fall far short of the subtle feats of location and identification that the human eye performs so effortlessly, these systems now suffice for a variety of important uses. A typical application is the identification of well-spaced parts moving along a conveyor belt; after being located, the parts can be grasped and placed in a standard orientation. Inspection of manufactured parts is another application growing in economic importance: here, the two-dimensional image of an object of moderate complexity, for example, a metal casting, is examined to make sure that all of its expected subfeatures, and no others, are present. In another recent application, visual sensing is used to find seams to be traversed by a robot arc welder; three-dimensional (structured-light) vision might be particularly appropriate in this application.

9.2.2 *Future Applications*

The object-inspection techniques mentioned in the preceding paragraph have been applied to inspection of faults in printed circuit boards, integrated circuit masks, and integrated circuits themselves. However, today's robot vision systems are too slow to be cost effective in dealing with images of this complexity. Falling computation costs and the development of special chips to perform basic image-processing operations at extreme speeds can be expected to remove this limitation and thus to expand the range of objects that can be inspected economically by robot equipment.

Still more sophisticated image-processing will be required before robots can move amid the clutter of the ordinary industrial environment. Initial experiments, such as those conducted at the Stanford Artificial Intelligence Laboratory, make the problem of unraveling views of ordinary, uniformly illuminated scenes appear quite challenging. Here, however, we can hope that three-dimensional vision methods will be superior; at any rate, the range of information generated by schemes of this kind should make it easier to locate and avoid obstacles. Deeper scene analysis methods will be required for still more sophisticated applications of vision; for example, locating and grasping objects partly hidden in the clutter of a standard industrial tote bin, or identifying trash cans standing near bushes and behind trees on a suburban lawn.

10.0 *OTHER RESEARCH PROBLEMS*

Although robotics research seems certain to touch upon a particularly broad range of technologies and scientific disciplines, some understanding of the areas likely to be significant over the next decades can be gained by surveying the near-term requirements of industrial robotics. These include:

(1) *Faster robots and increased robot accelerations.* The productivity of robot equipment depends upon the speed with which it moves. For this reason, the seemingly pedestrian problem of increasing the rates of motion and acceleration of robot manipulators is of particular economic importance. This is partly a problem in mechanical engineering. It is also a matter of combining a more sophisticated understanding of the elastic reactions of rapidly moving manipulator members with the computer control techniques to compensate for the mechanical and sensory inaccuracies of high speeds and accelerations and to suppress unwanted oscillations. The solution will require improved force sensing and more sophisticated servocodes.

(2) *The improvement of optical, tactile, and other robot sensors is an important near-term research goal.* Tactile sensing plays a particularly important role in dexterous manual assembly. The subtlety of the human tactile sense is far from being matched by the relatively crude sensors currently available with robot manipulators. It seems clear that greatly improved sensors will be required if complex assemblies, especially of fragile and deformable parts, are to be attempted, if robots are to work extensively with soft or plastic materials such as wires, cloth, or vinyl, and if more sophisticated methods of grasping are to be developed. Important work under way on tactile sensors should yield considerably improved sensors within a few years. These sensors will be able to detect not merely that a robot's finger is touching something but, also, the precise parts of the finger at which contact occurs; in addition, they will be capable of delivering at least a crude "tactile image" of the surface touched. (To see how important this is, note carefully what you do when turning a page in a book, especially with your eyes closed. A delicate tactile sensation of the manner in which the edge of the page being turned rests against the operative finger is necessary in order to slip the finger under the edge of the page at just the right time to accomplish turning.)

It is also important to develop improved proximity sensors capable of giving advance warning of impending collisions. Without adequate proximity sensors, one will never be willing to set robot arms into rapid motion in an environment that is to any degree unpredictable. It is plain that the development of better proximity sensors can contribute substantially to the productivity of robot systems and also to their flexibility. In this domain, there is a curious absence of deep analysis of the acoustic signal returned from a sonic wave impinging on a complex scene. Roboticists have taken a superficial view of this problem, examining a returned wave for time of first arrival only, and neglecting its deeper structure.

Better tactile and other sensors will, in turn, call for more sophisticated software to manage them, a consideration that emphasizes the significance of improved programming techniques and higher performance computers to the general progress of robotics.

(3) *Force-controlled motion primitives.* The motion control primitives supplied with today's robot systems are purely geometric in character, but cannot, as they stand, be used to cause a robot arm to move smoothly while it maintains contact with a curved surface of unknown shape. This ability is essential for the smooth and logically flexible adaptation of a manipulator to an environment in which the whole geometry is not known in precise detail.

Force-controlled motions are essential in manual assembly; blind workers can function in many factory roles where a worker with anesthetized hands might be useless. The demonstrated advantages of devices like the Draper Laboratory's Remote Center Compliance, referred to

earlier, illustrate this. As anyone who has struggled to close a tightly fitting desk drawer knows, a long object being pushed into a closely fitting cavity can jam easily. Using geometric analysis, it is easy to see that the tendency to jam is greatly reduced if the object is pulled into the cavity rather than pushed. Of course, this is normally impossible: one could hardly crawl into a desk to pull a drawer in from its reverse side. However, the simple but ingenious arrangement of hinges used in the remote center compliance results in exerting exactly the pattern of forces that normally would have to be exerted by pulling the body from its other side. The wide industrial interest in this device points clearly to the importance of proper force control for robot manipulations. Research and development efforts that build on the success of the remote center compliance and its underlying theory are therefore likely. Ultimately, they should make force-controlled motion primitives available in the commonly used robot programming languages.

(4) *Geometric modeling.* Geometric modeling refers to the use of a computer to build models of three-dimensional bodies and the constraints on their relationships. These then can be combined into a comprehensive environment model within which various kinds of kinematic and physical analyses become possible. These "computer-aided design" models are used fairly widely by designers to produce graphic images of an object being built, to generate engineering prints, and, in particularly advanced systems, to build control programs for the numerically controlled machine tools which produce the object that a designer has conceived.

The demands of robotic applications can be expected to force systems of this type to considerably higher levels of sophistication. For example, graphic systems may play an essential role in debugging robot control programs. Especially if multiple arms and complex synchronized motions are involved, attempting to debug control programs by actually setting expensive machinery into motion may simply be too dangerous because of the possibility of collision. It would certainly be preferable to construct a purely graphic world in which one could view the motions that a control program would trigger. However, generating computer motion pictures of complex curved bodies in the volume required for this type of debugging will require geometric procedures of an efficiency and sophistication considerably beyond what is possible today.

Modeling the motion of bodies in contact, as influenced by their geometry, elasticity, and mutual friction, is a related but even more challenging problem. Ultrafast computers may be required.

A key problem in robot modeling, but one for which no results whatever have yet been obtained, is how to update an internally maintained geometric world model when an unexpected collision occurs during the execution of a planned path.

(5) *Computation complexity of robotic calculations.* Pragmatic control and sensing problems currently being encountered in robotics are complex and require much further work. Many will undoubtedly be solved. On the other hand many of the complexities arising in the planning of motion are provably intractible, and hence one must carefully distinguish between tasks that are possible and those that are impossible. This may be particularly important in problems in three-dimensional motion involving interactions between multiple objects. Knowledge of the computational complexity of these problems can serve as a useful guide to the practitioner. It also suggests that no single unified approach may be able to handle all of these problems, so that a large amount of practical detailed work and the integration of varied approaches will be require.

(6) *Robot locomotion.* Robots with legs (or wheels, or half-tracks), which are able to move through their working environment, are desirable for a variety of applications. One interesting case might be a "brachiating" robot that clambers, monkey-fashion, over the surface of a spacecraft (or submarine vessel) to make repairs, or over a space station it is building.

Current applications of self-moving robots are largely rudimentary, and the few interesting robots of this type that have been built exist only in research laboratories, where they have been used to investigate some of the many problems connected with robot locomotion. One of these is avoiding obstacles and finding paths in an environment not known ahead of time; aside from its planning aspects, this is basically a problem in visual analysis and range-sensing.

The problem of legged locomotion has been studied by groups in the Soviet Union and the United States. The most active U.S. group is that at Ohio State University, where researchers have constructed a moving "hexapod," over which experimental control is exercised by a simple "joystick" used to elicit forward, sideways, or turning locomotion. As of 1982, the Ohio State work has reached the point at which the hexapod can stride successfully over a flat floor at a speed of 20 feet per minute; work on sensory adaptation to an irregular terrain is beginning.

A group at Carnegie-Mellon University has begun studying the more complex problem of dynamic stabilization of statically unstable walking robots. As these basic kinematic/dynamic studies achieve success, robot "leg" systems will become available. The "legs" will be controlled relatively simply, for example, by geometrically specifying some desired path over terrain, and also by defining the manner in which the robot is to adapt to terrain irregularities, the way in which proprioceptive "balance" signals indicating incipient falls are to be handled, and so on. Like all other primitive robot capabilities, this should result in the addition of appropriate new statements to the languages used to program higher level robot activities.

The program controlling a self-moving robot will have to keep track of the robot's constantly changing position and use this information to handle references to objects fixed in the robot's environment. Although known geometric techniques can be used, no commercially available robot programming system offers this feature.

(7) *Improved robot programming techniques.* A considerable body of literature on industrial assembly gives detailed directions for producing a great many common manufactured items. Some way of automatically translating these manuals into robot assembly programs would be ideal but, unfortunately, this objective far exceeds the capability of today's robotics programming languages. For anything close to the language of standard industrial assembly manuals to be accepted as robot control input, much more sophisticated languages will be required. The compilers of such languages will have to incorporate knowledge of the part/subpart structure of partially

assembled objects, as well as routines capable of planning the way in which objects can be grasped, moved without collision through a cluttered environment, and inserted into a constrained position within a larger assembly.

This level of programming sophistication only becomes feasible if a robot system can maintain either a detailed model of the environment with which it is dealing, and keep this model up to date through a complex sequence of manipulations, or if the robot can acquire and refresh such a model through visual and tactile analyses of its environment. Although no robotic language with this degree of sophistication has been produced, such languages have been projected, for example, in the work on AUTOPASS at IBM, and its Stanford University, Massachusetts Institute of Technology (MIT), and University of Edinburgh relatives, AL, LAMA, and RAPT.

It should be noted that the implementaion of such sophisticated languages will require the solution of many complex mathematical and geometric problems. A basic one is that of planning collision-free motions of three-dimensional bodies through obstacle-filled environments. This problem, studied by researchers at the California Institute of Technology, IBM, MIT, New York University, and elsewhere, has been brought to a preliminary stage of solution but, from the practical point of view, the work merely reveals the complexity of the computations that motion planning involves and the importance of seeking much more efficient motion-planning schemes.

Work reported by MIT and other laboratories also suggests possibilities for more advanced software that plans robot activity. Working within a simulated world of blocks, MIT researchers constructed a program that could combine geometric knowledge of the collection of blocks given it with an understanding of the desired final assembly to produce a fully sequenced assembly plan. This demonstration program was even capable of using some of the blocks available to it to construct fixtures useful in the assembly of the remaining blocks.

(8) *In-depth studies of important current application.* Spot welding by robots has become routine, and attention is turning now to the more complex physical problems associated with continous arc welding, where proper control of welder robots requires some understanding of the thermodynamics of the liquid-solid arc pool. Ways of developing specialized robots to work in environments hazardous or otherwise inaccessible to humans, such as high-purity clean rooms, deep-sea environments, nuclear reactors, and space, also require detailed study and will sometimes raise very complex dynamic and other problems. Improving robot tactile sensors and motion control software to the point at which robots could work successfully with soft industrial materials, for example, leather, foam rubber and/or vinyl furniture coverings, is of obvious economic importance to such industries as furniture and shoe manufacture.

11.0 *FUTURE APPLICATIONS*

(1) *The Household Robot*

Even the relatively simple problems of industrial robotics are quite challenging, and we are far indeed from being able to create useful, general-purpose household robots. Nevertheless, this intriguing and often-discussed possibility is worth examining--not because such applications are urgent, but simply to form some impression of the technical problems that would have to be faced in order for household robots to become feasible. In a sufficiently advanced technology, one might hope to apply robots to ordinary household tasks: cleaning, returning items to their storage positions, unpacking groceries, vacuuming, dusting, folding laundry, making beds, and so on.

Major technological advances will be required before robots can perform familiar tasks. The household environment is much more varied than the industrial environment, and it is not nearly as controllable. Rather than being able to deal with a very limited number of workpieces whose geometries are known in detail, a household robot would have to handle objects of many shapes and sizes. In a household environment, these details would change from day to day. Thus, for the household use of robots to become practical, the level of robot command languages and sensing will have to be raised far beyond the requirements of industrial robotics. Such a robot would have to be able to handle natural language inputs, implying a considerable understanding of the household environment, greatly improved visual recognition capabilities, and the ability to move easily and safely amid clutter.

(2) *Other Future Applications*

Although general-purpose household robots appear infeasible at present, it might be possible to design robots to perform a broad spectrum of other social "housekeeping" tasks. For example, robot sweepers for large public spaces, such as streets, sports stadium, railroad stations, airports, etc., are probably within the reach of current technology. Robot lawnmowers, with programmed awareness of the boundaries of a grassy area, also seem feasible. A related possibility is computer-controlled agricultural equipment for automatic plowing, cultivation, and reaping.

Greater mastery of geography and geometry, based on more sophisticated sensing and locomotion, might permit robot mail, package delivery services, and trash collection. Greatly improved visual and other sensors, together with much more highly developed robot programming techniques, might make robot automobile maintenance, and even some degree of repair, possible. Significantly improved tactile and visual sensing procedures, together with additional theoretical understanding of the behavior of such soft materials as cloth, might allow the development of robot tailors, which could use an appropriate vision algorithm to sense the contours of a client's body and, from this, produce a suit of clothes.

By feeding control signals taken from a human activity into a robot control program, it should be possible to develop various semirobot prostheses, for example, sophisticated gripper, for the armless.

12.0 *TRAINING A GENERATION OF ROBOTICISTS*

We expect the requirements of robotics to make a significant reorganization of the present computer science curriculum necessary, since robotics research will need to make use of a much wider range of classical mathematics and physics than it has been involved with until now. This includes computational algebra, computational geometry, servomechanism theory, mechanics, theories of elasticity and friction, materials, science, and manufacturing technology. Mathematics, physics, and engineering departments will be in closer contact with computer science than ever before. The complexity and high content of classical science may make robotics into a recognized professional

subspecialty within computer science.

Development of the field of robotics will require increased numbers of researchers. Their education will of necessity cross traditional lines between computer science, mechanical engineering, control theory and material science but the central area is undoubtedly computer science oriented. For universities to develop programs to adequately educate these researchers will require two things. First, what is needed is a conceptualization of knowledge that comes from abstracting and studying precisely stated problems capturing the essence of complex engineering tasks. Second, research facilities are needed to experiment and uncover underlying problems that would be overlooked in purely theoretical studies. Many of the interesting problems in robotics arise only in trying ideas in practice and realizing that what was thought to be trivial aspects of a task involve fundamental research.

12.1 *Applications Oriented Education.*

There is also a requirement, arising from the need to deploy robots rapidly to American industry, to generate a course in the application of robots. The body of knowledge required to design and manage assemblies and other robot manufacturing applications is very wide, but does not necessarily reach very deep into the many domains with which it makes contact. For example, a successful assembly application may require a rudimentary knowledge of material strength, of jigs and fixtures, of the functional phenomena of close-tolerance assembly, a knowledge of time and motion analysis techniques as they apply both to human and to robot assembly, etc. A knowledge of effective precedence in assembly strategies is also useful. Some of this information is beginning to be captured in a body of experimental subroutines which have evolved in connection with common assembly tasks such as putting pegs into holes; this knowledge needs to be put on public record and disseminated.

Accordingly, the definition of an initial course in assembly techniques could be a useful output of an NSF-sponsored activity. Collection of a systematic videotape library showing various important automatic machines and devices, processes, human and automatic manufacturing operations, etc. would give useful support to such a course. These tapes could convey very useful information, provide motivation, and define a standard against which the applications engineer could test his own expertise.

Vocational training in what maintenance will have to be handled by a wide range of educational establishments, supported by industry and state/local government, as is normal in other technological areas.

12.2 *Facilities.*

The initial costs for a research group entering robotics are substantially higher than those needed to enter other areas of computer science and engineering. The equipment needed is more specific to individual projects and to individual researchers, as opposed to expensive general computing facilities. Furthermore, robot equipment requires mechanical maintenance and support. As more and more universities enter the field, a special equipment program may therefore be necessary, but it is not clear whether such a program is economically feasible. Minimal costs for a precise industrial-quality arm are $40,000, plus the cost for site preparation and technical support. Minimal grants may therefore have to be in the $200,000 per year range. To overcome the obvious difficulties of equipment dissemination in this area, various alternatives and options exist which might make it possible for a facility to acquire direct experience with input equipment. Interdisciplinary settings should be strongly emphasized. Other possibilities are organized summer programs at institutions such as CMU or the National Bureau of Standards, which possess major facilities. More systematic organization of industrial equipment grants may also be desirable. Organization of regional consortia for sharing of experimental equipment is another possibility. Government funding agencies are encouraged to consider this area carefully.

13.0 *THE SOCIAL IMPACT OF ROBOTS*

To the extent that hazardous, repetitive, and menial tasks are taken over by robots, and to the extent that robots contribute to U.S. productivity and international economic competitiveness, we can take satisfaction in this technology; it simply continues (though significantly generalizing) the trend to increased automation that has characterized the American economy for the past hundred years. On the other hand, some cautionary reflections are suggested. The gradual unfolding of the enormous potential of artificial intelligence can be expected to affect profoundly the fundamental circumstances of human existence. Today's robotic developments exemplify this. Although it would be wrong to forget that much challenging technology must still be put in place before robots can reach their full potential, it is nevertheless true that, as robotic science develops, it will progressively reduce the labor used in industrial production, ultimately to something rather close to zero. If society can respond appropriately to this deep change, we may profit in the manner foretold by Aristotle: "When the Loom spins by itself, and the Lyre plays by itself, man's slavery will be at an end." If, on the other hand, we fail to adjust adequately, for example, if the growth in service industries that has characterized recent U.S. economic history proves insufficient to absorb all of those persons likely to be forced out of industry by the development and installation of robot equipment, grave social tensions may develop. Assessing these difficulties and dealing wisely with them will be important tasks for both social thinkers and national leaders.

14.0 *OUTLOOK*

Although the sensory and manipulative capabilities of industrial robots and their ability to handle unexpected events are still quite limited, robot technology is strengthening and expanding rapidly. Initial commercial success has been achieved through robot mastery of such commercially important operations as spot welding and spray painting. The microelectronic revolution can also be expected to accelerate the development of robotics by supplying many "miracle chips" embodying advanced sensory and control functions.

While all robots will only come to perform the majority of industrial operations gradually, their potential near-term uses are extensive. The area most favorable for robot application at the moment is the manufacture of items in batches substantial enough for hand manufacture to be undesirable but not large enough to warrant heavy capital outlays on fixed automation. Robotic research is proceeding actively in France, West Germany, Sweden, Japan, and the United States, but Japan leads in the use of

robots. Operating industrial robots in 1980 number 11,250 in Japan and 4,370 in this country.

Much of the robotic equipment used by industry today is too crude to respond flexibly to changing situations. Much more sophisticated control languages are needed to make industrial robots easy to use. Desirable new robot language features include, for example, statements that can cause a manipulator arm to move between specified positions while automatically avoiding certain obstacles or undesirable configurations. In general, we can expect advances in robotic technology to be reflected by the addition of new statements to robot programming languages.

More sophisticated sensory functions, e.g., improved image-analysis software, represent another important direction. These sensory capabilities will need to be integrated with manipulator-control software, to produce useful industrial hand-eye systems.

At present, software for analyzing visual data acquired by robots is not especially reliable and is quite expensive computationally. Improved scene-analysis algorithms are needed if we are to have vision systems that can easily be reprogrammed for a wide range of applications. Deeper understanding of the algorithmic nature of such systems will encourage development of high-performance electronic chips that can handle the required operations very rapidly. As such chips become available, the range of objects that can be inspected economically by robot equipment will expand.

The limited present capabilities of industrial robots indicate that we are far from being able to create general-purpose capabilities far exceeding those needed by industry. Still, it may soon be possible to design robots to perform relatively simple housekeeping tasks, such as sweeping large public spaces or mowing lawns. A related possibility is computer-controlled agricultural equipment, such as plows and cultivators.

Robots can be expected to continue to take over hazardous, repetitive, and menial tasks and to contribute significantly to the progress of industrial automation under way in this country for the past century. Moreover, robot technology can be expected to reduce the industrial labor force very drastically. In the absence of adequate mechanisms for adjusting to this change, severe social tensions may result. Our national leaders face the critical task of assessing and forestalling these developing problems.

BIBLIOGRAPHY

Hiroyuki Yoshikawa et al. *Computer-Aided Manufacturing: An International Comparison.* Washington, D.C.: National Academy Press, 1981.

James L. Nevins and Daniel E. Whitney. "Computer-Controlled Assembly," *Scientific American,* Vol. 238, No. 2 (February 1978).

Robert Ayres and Steven Miller. *The Impacts of Industrial Robots.* Robotics Institute, Carnegie-Mellon University, 1981.

APPENDIX A

APPLIED ARTIFICIAL INTELLIGENCE

An Assessment of the State-of-the-Art and
Recommendations for Future Directions

Prepared for the Information Technology Workshop
Xerox International Center
Leesburg, VA
January 5-7, 1983

Abstract

This report covers three main AI application areas: (1) natural language systems, (2) expert systems, and (3) support software for these types of applications, most notably knowledge representation languages. Each area is treated separately. The discussion of each area includes an assessment of the state-of-the- art, an enumeration of problem areas, an enumeration of opportunities, recommendations for the next 5-10 years, and an assessment of the resources required to carry them out. Also included is a discussion of possible university-industry-government cooperative efforts.

1. *Introduction*

A recurrent theme in this report is that progress in both natural language and expert systems depends heavily on building knowledge bases for different bodies of world knowledge. We now have at least a few moderately successful programs that can sit on top of an appropriately designed knowledge base and either allow a user to ask questions about the knowledge base in English, or (using a different type of knowledge base) do "expert" reasoning of a sort about event instances of the kind described in the knowledge base. Thus it seems useful to separate research into *processing components* such as natural language parsers and semantic interpreters and expert system "inference engines", and *knowledge bases*, that is, suitably encoded bodies of knowledge on which and with which the processing components can operate. In recent years, attention has steadily shifted toward problems of *knowledge acquisition*, that is the building of knowledge bases to be used by other systems. While there has been work on natural language and expert systems that can automatically extend their own knowledge bases, we will assume in the first portion of this report that for the near future, the problems of processing and knowledge base building are separable.

There are reasons for and against this point of view.

Pro:

It is good to separate general purpose knowledge and procedures (e.g. parsers or "inference engines") from domain dependent knowledge and procedures, so that new applications are made simpler.

Modular programming is a good idea in general.

Con:

Radical separations of general and domain-dependent program portions are artificial and difficult to achieve because there are so many borderline cases, and because the kinds of representations we choose affect the kinds of processing we have to do.

Modular solutions encourage us to pick simple representation schemes ahead of time which may later prove to be awkward to use. For example, production systems have very nice modular structure, but present difficulties if one is trying to represent procedural knowledge (which can lead to the extensive use of productions whose right-hand sides merely "call" other productions) or metaknowledge (which should ideally select or order a group of productions, but can lead to very large left-hand sides, specifying the context of applicability of the rules).

2. *Natural Language Systems*

For the purposes of this report, we will also make a radical separation between work on practical natural language systems and theoretical work on natural language understanding that is not directed toward any particular application. Clearly, the boundaries of this separation will change over the next ten years; topics that are now only dealt with in a theoretical setting will become candidates for practical applications. We will attempt to give estimates of how relatively difficult it will be to master various areas.

2.1. *An Assessment of the State-of-the-Art in Natural Language*

2.1.1. *The Players*

Let us begin with a listing of the institutions that figure prominently in natural language (NL) research and development.

2.1.1.1. *Companies*

The following companies offer commercial NL products (or soon will):

AI Corp., Waltham, MA. (Larry Harris) The first to hit the marketplace with ROBOT, an NL data base front end, now supplanted by INTELLECT. These systems are based on a fairly general parser, which tries out a number of possible parses on a fully inverted database, keeping those interpretations that have reasonable numbers of "hits" (i.e., non-zero but much smaller than the whole database). Can be installed in a couple weeks with local talent, given a suitable database and reasonably sophisticated local manager.

Cognitive Systems, Inc., New Haven, CT. (Roger Schank) Offers a range of NL DB systems, and NL expert systems, though only a handful have been installed to date. Program mechanisms not yet made clear, but seems to based on a semantic grammar of sorts. Capable of handling ellipses and some non- grammatical input.

Symantec, Sunnyvale, CA. (Gary Hendrix, Ann Robinson) Plans to offer NL systems evolved from the LIFER and KLAUS systems at SRI, but scaled down for use on microcomputers. Systems will also offer input and access via forms (query by example).

Hewlett-Packard, Palo Alto, CA. (Ira Goldstein, Fred Thompson) Has potent NL DB systems evolved from Thompson's REL work.

Texas Instruments, Dallas, TX. (Harry Tennant) Has some NL systems. Unknown whether or not TI has marketing plans.

The following industrial research labs have done extensive NL system work, but do not seem likely to market products in the near future.

Bolt Beranek and Newman Inc., Cambridge, MA. (Bill Woods, Rusty Bobrow, Candy Sidner) A wide variety of work, most notably the LUNAR NL DB system, the HWIM (Hear What I Mean) speech understanding system, KL-ONE knowledge representation system, and RUS NL parser.

SRI International, Menlo Park, CA. (Nils Nilsson, Barbara Grosz, Bob Moore, Don Walker, Kurt Konolige, Jane Robinson, others)

Again, a wide variety of work, including the LIFER NL system for accessing distributed data, KLAUS and TEAM, more recent and much more powerful NL systems based on general NL parsers. Also work on extensions of logic to allow commonsense reasoning for NL systems.

Xerox PARC, Palo Alto, CA. (Danny Bobrow, Terry Winograd, many others) Research on knowledge representation, programming environments and tools for AI system development, especially INTERLISP, and more.

IBM Thomas J. Watson Research Labs, Yorktown Heights, NY. (George Heidorn, Warren Plath, Stan Petrick, Fred Damerau, others) Two separate groups at work here. Heidorn's group has EPISTLE, a dictionary-based system designed to critique test, using an extensive parser, with hopes to expand applications areas. Plath's group has developed TQA, a transformational grammar-based DB system that has already had extensive field testing with quite impressive results.

Fairchild Advanced Research Laboratories, Palo Alto, CA. (Peter Hart, Ron Brachman, Phil Cohen, Hector Levesque, others) Research especially in knowledge representation and pragmatics, that is, representing and understanding the goals of language use.

2.1.1.2.*Universities*

The following universities are active in NL:

Yale (Roger Schank, Chris Riesbeck) Research covers text and story understanding, knowledge representation, inference from NL, cognitive modeling, and other areas.

University of Pennsylvania (Aravind Joshi, Bonnie Lynn Webber, Tim Finin, Ellen Prince, Tony Kroch, others) Extended NL interactions (monitors recognizing and reconciling user misconceptions, explaining DB concepts, justifying responses), NL generation and interactive help systems.

University of Illinois, Urbana (David Waltz, Jerry DeJong, Jerry Morgan, Bill Brewer, Andrew Ortony) Research on schema learning through language, story understanding, metaphor, plausibility judgement, representing time, parallel excitation and inhibition models of meaning selection, and more. Past work on NL database front ends.

Carnegie-Mellon University (Allen Newell, Jaime Carbonell, Raj Reddy, Phil Hayes, others) Work on learning, metaphor, practical NL interfaces to computer systems, speech understanding and general architectures for AI systems, and more.

University of California, Berkeley (Bob Wilensky, Lotfi Zadeh, George Lakoff, Charles Fillmore, ...) Research on NL interfaces to UNIX, use of planning knowledge in understanding text, fuzzy logic and representation of meaning through "test score semantics"; good AI-oriented linguistics work on campus.

University of Massachusetts (Dave McDonald, Wendy Lehnert, Edwina Rissland, Michael Arbib, Barbara Partee, others) Emphasis on adaptive network models for AI, story understanding, modeling of emotions and "narrative units", learning, Montague grammar, language generation, and more.

Stanford University (Terry Winograd, John McCarthy, Gordon Novak, others) Work on knowledge representation, language generation, especially for expert systems, nonmonotonic logic and commonsense reasoning.

University of Texas, Austin (Robert Simmons) Research over the years in semantic network representations for NL, understanding physics problems stated in NL, linking language and perception.

University of Delaware (Ralph Weischedel) Research on developing NL system-building tools, practical NL interfaces.

NYU (Naomi Sager) A large transformational language system for a medical data base. Probably the most complete grammar available.

USC Information Sciences Institute (Bill Mark, Bill Mann, others) Research on NL tools, knowlege representation systems, pragmatics, NL generation, and more.

Brown University (Gene Charniak, ...) Knowledge representation, inference mechanisms, integrated syntactic and semantic NL parsing.

University of Rochester (James Allen, Pat Hayes, Jerry Feldman, Steve Small, others) Research on representation of time, knowledge representation especially using predicate calculus, "connectionist parsing", and more.

University of Connecticut (Rich Cullingford) Use of scripts for NL representation.

Columbia University (Mike Lebowitz, Cathy McKeown) Representing NL, understanding patent descriptions, other work.

Work overseas has not been listed here, though there is also research going on in Canada, Germany, Japan, France, United Kingdom, the Netherlands, Sweden, Poland, Czechoslovakia, and Italy, in rough order of apparent effort.

2.2. *An Enumeration of Problem Areas in Natural Language*

The central difficulty here is that there is no such thing as a small natural language system. If a system is to accept natural language, that is, what people naturally think of

saying in the manner they think of saying it, it must have a large vocabulary, a wide range of linguistic constructions, and a wide range of meaning representations. A small system cannot be very natural.

Overall, we believe that we have a very long way to go before we will have truly general-NL systems. Knowledge representation is a key problem, and is tightly coupled to the problem of deciding just what knowledge should be included in a system. All systems to date have depended on having a restricted domain in which to operate, so that the number of ambiguities that remain after all the knowledge of the system has been brought to bear can be manageably small.

Furthermore, most NL systems have been built to accept only highly restricted classes of "speech acts". Thus, data base question answerers usually assume that everything that they are given falls into one a few classes such as request for data, request for help, or data exclusion statement, and story understanding programs assume that all the language they get falls into the classes of story fragment or questions about a story. In general, of course, language may be used to accomplish many other kinds of ends: propaganda, exaggeration, advertisement, humor, poetry, education, correction, sarcasm, irony, lies, etc. A system that is designed to handle unrestricted text, for example transcripts of conversations or contents of books, can not make these kinds of simplifying assumptions, and must be prepared to understand and represent a wide range of possible motives for various types of language.

Even more problematic, in order to judge what types of language they are being given, systems must have robust facilities for "plausibility judgement", that is, they must be able to judge whether the sentences that they are given can make literal sense, or whether they must be interpreted as metaphor, lies, exaggeration, etc. This requires (1) a body of "common sense" knowledge of the physical world as well as of the world of people and behavior, and (2) a good grasp of the ordinary ways in which ends are achieved with language, so that stereotyped speech acts can be recognized as can novel or unusual uses of language. We are very far from an understanding of this sort.

Certain kinds of language have been given very little attention: language that describes the physical world and operations on it (such as instructions for building mechanisms or text that accompanies a diagram); reports of inner experience or feeling (such as a patient's verbal medical history); real conversations or arguments.

The types of computers that are available today seem to be rather poorly matched to most of the models of human language understanding that have been advanced in recent years. For example, it is quite well-established that the more ambiguous a sentence, the more time required to process it, even though the person understanding the sentences has no conscious awareness of ever considering any more than one interpretation for each sentence. This suggests pervasive parallelism, which is particularly difficult to program on current hardware. The only types of parallelism we have had much success with have been those where we can have uniform arrays of processors each working on rather local computations. Language seems to require parallel consideration of a variety of global possibilities. The upshot is that we have almost certainly placed an overemphasis on serial language understanding algorithms that may have very little to say in the long run about human language processing. There is a chicken-or-egg type problem here also: it is not yet possible to say exactly what type of parallel hardware we ought to have to do NL properly, and until we have proper parallel hardware, parallel NL programs run on serial machines may not run fast enough to be reasonably tested or to hold any interest for those who want new and better application programs. Special algorithms evolved from ones in use today will probably have little long term interest, and may suffer in generality because they are forced to make choices without fully considering all possibilities.

Continuous speech understanding is still rather fragile: speaker-dependent, intolerant of noise or accents, and capable of only a small vocabulary. It also requires either a large and expensive processor, or much longer than real time to process speech.

We also need new design methodologies. Even restricted NL systems can be very large, and since the success of an approach can not really be measured until a system embodying it is completed and evaluated, the design-test-redesign cycle tends to be long. The development of a full scale system is also a problem. Let us suppose that we want a system with a 10,000 word vocabulary; while a portion of a lexicon for the NL system can be extracted from a dictionary, each definition also requires seperate attention, since dictionaries do not contain all the information necessary (definitions tend to be circular). If a team approach to construction is used, the team must be small in order to avoid the types of problems detailed in "The mythical man-month" [Brooks 1977?] -- as a programming team gets larger, its members tend to spend all their time talking to each other about standardization, intermodule communication, updates, etc. and produce fewer and fewer lines of code per person per day. On the other hand, if a programming team is very small, it will simply take a long time to produce the necessary code because of individual programming speed limitations. In addition to programming aids, designs that are highly modular can help by dividing a problem up into relatively small independent pieces.

There are special practical problems in advancing NL science and technology at universities. Progress is slowed because Ph.D. candidates are expected to work independently; the design and building of any significant NL system is a massive undertaking, and one that generally can not be split and still allow suitable academic credit to be assigned to all participants. In general, system design is a necessary part of the research and it is not easily possible to evaluate designs without implementations. Once a nice system has been shown to be feasible, there are further difficulties; universities are not in a good position to develop or support software, since the designer/builders generally leave after receiving a degree, and there are few academic points awarded for cleaning up someone else's system and making it robust.

Industries have special problems as well. The largeness of NL systems, and the fledgling state of our current technology makes it difficult or dangerous to promise profitable products on any time scale short enough to be appealing to management. There is also a wide gap between many of the kinds of programs that are produced in academia and the kinds of programs that can become good products. For example, while story understanding programs are

considered (rightly) an important current academic topic, such programs have no clear short term industrial potential.

The comments in this section so far have assumed that we will continue to do research in roughly similar ways to those that have been employed in the recent past, that is that we will attempt to build AI programs that are "instant adults", systems which are the result of programming, not of learning. There has been a growing interest in learning over the last few years, although relatively little of the learning research has been directed toward NL. Certainly there are serious difficulties with the engineering of a system capable of learning language from experience (an "instant infant") -- the most serious problem seems to be adequate perceptual apparatus for extracting "important" items from raw sensory data. Even if we could devise such a learning program, there is little reason to suppose that it could be programmed to learn language much more rapidly than a human can, which would mean that at least several years would have to pass before we could judge whether or not the system was adequate. And, of course, the chance of getting everything right the first time is close to nil. Nonetheless, this seems to be an important avenue to explore, to hedge our bets, to further cognitive science, and perhaps to find a compromise position ("instant five-year-old"?) that would represent the optimal long-term route to fully general NL systems. Further reasons for emphasizing learning include the observations that the only language users that we know are built through learning, and that we continue to use learning as adult language users, so that we probably need learning of some sort in our programs anyhow.

Another practical problem is finding good application areas, that is, ones where NL is truly helpful, where the domain is well-circumscribed and well-understood, where the tools that we now have are capable of conquering the problem, and where typing is possible and acceptable as an input medium.

Finally, there is an acute shortage of qualified researchers. Since most actual NL application programs are likely to be the result of development by industry, it seems desirable to encourage more industrial effort. However, taking researchers away from universities reduces the number of faculty capable of supervising graduate students and carrying on much-needed basic research.

2.3. *An Enumeration of Opportunities in Natural Language*

Once continuous speech understanding is possible, we believe that the number of good NL application areas will grow dramatically. Continuous speech systems seem close enough to reality (commercially available under $20,000 within 3-5 years?) that we should begin to think more seriously now about the realm of new application areas that speech systems will open up. It is important that basic NL understanding research continues so that we will have adequate tools and understanding of the nature of the fundamental problems in NL, so that we can then move rapidly to produce useful systems.

We now have available a variety of parsers and knowledge representation systems, which should make it possible to build new systems more rapidly, by using off-the-shelf components for programs. We do need to have these systems well-documented, and we still need more experience in tailoring systems from these components, but there is the opportunity to begin building custom NL systems now, at least for limited task domains. Research and development of good NL tools should pay excellent dividends.

It is now possible to think about a much wider range of types of NL processing because machines have become substantially larger and address spaces have become reasonable for NL systems. In the past, a great deal of effort was devoted to attempts to collapse code length so that space could be made available to expand the capabilities of NL programs. This is no longer such a problem, and we can trade off space for ease of programming. In addition, it is now possible to relatively easily produce special-purpose chips, so that, for example, we can realistically consider algorithms for highly parallel word sense selection, truly concurrent syntactic, semantic and pragmatic evaluation of sentences, speech format extraction, etc.

A number of good application areas are now possible. Examples include NL data base front ends; NL interfaces for expert systems, operating systems, system HELP facilities, library search systems, and others software packages; text filters, text summarizers; machine-aided translation; and grammar checkers and critics.

Many good future applications are possible, especially in conjunction with "information utilities" (i.e. information services that are available via phone or cable connections). Possible public services include automatic directories of names, addresses, yellow pages, etc.; electronic mail; on-line catalogues and ordering facilities, banking and tax services; routing directions (like AAA); access to books and periodicals, through titles, authors, topics, or contents; and undoubtedly many others. All of these services could also have on-line NL help facilities and manuals. There are parallel needs in business and in the military services, e.g. for command and control, inventory control, ordering and shipping, coordination of organizational planning and plan execution, and so on.

Another good application bet is instructing robots in NL, much as one would instruct a human assistant. This can allow the robot to understand the goals of the instruction, rather than just the means for achieving the goals, as is the case now with teaching-by-leading-through-motions or "teaching" by ordinary programming. Understanding the goals can help a robot in dealing with a wider range of situations and for recovering from errors.

2.4. *Recommendations for the Next 5-10 Years in Natural Language*

1. Continue a broad range of basic research support; there is still a shortage of science on which to base NL system engineering.

2. Encourage cooperation between AI NL research, and the traditional fields interested in language (linguistics and psychology). This can serve the purpose of aiding the building of a science of language and cognition, and can also provide a more rapid increase in manpower resources than is possible through the training of new researchers only. This recommendation may happen anyhow, since "social science" funding has been cut dramatically, putting pressure on many of the kinds of people that could help the AI NL effort. To make this work, a fair amount of re-education, especially in

AI, computation and programming, will be needed, along with some re-education of AI people in linguistics and psychology, to build a common knowledge base.

3 Increase funding for equipment. Adequate equipment is essential if people are to produce working systems, rather than just theoretical advances. Increasing available compute power can dramatically cut the amounts of time and effort required to produce running systems, since easy-to-write though computationally expensive systems can then be considered. Squeezing a large program onto a small machine can be very time-consuming. And if programs take too long to run, programmers start to work on speeding them up instead of working on increasing their range of competence. Despite the apparently high initial cost compared to salaries, money spent on hardware is likely to be a good investment.

4 Create and encourage development groups in industry and military labs, and encourage increased contact between such groups and university and industrial basic research laboratories. Universities are particularly ill-suited for development efforts, since there is a high turnover of system builders, making it difficult to support application systems. In addition, we need all the effort on basic problems that can be mustered. Development could be handled by groups that have more traditional software backgrounds; once feasibility has been demonstrated, AI systems often look a lot like other programs.

2.5. *An Assessment of the Resources Required in Natural Language*

The main needs are manpower and equipment. Except for very limited special purpose applications, NL systems require relatively large program address space and relatively large numbers of machine cycles. When done on the time shared machines in general use (e.g. Vax's and DEC-20's), research is clearly hampered by a shortage of machine power. LISP machines are much better, but some (e.g. Xerox Dandelions and Dolphins) have rather small address space, and the rest tend to be quite expensive if dedicated to a single user. Realistically, about $25K (approximately the price of a Xerox Dandelion) per researcher is needed to provide at least a semblance of the best environment for NL research; anything less will mean that progress is slower than it could be. In this estimate, we have assumed that a number of researchers already have such machines, and that some researchers will get more expensive machines (Symbolics 3600, LMI Nu machine, or Xerox Dorado). Including graduate students, research programmers, as well as senior researchers, there are now probably 300 people doing NL research. This brings the total expenditure needed to properly equip NL researchers for maximum progress to about $7.5M.

Manpower needs cannot be solved quickly by simple infusion of dollars. Money that is spent on university equipment will probably do the most good, because it can help speed the research for graduate students, and it can also make universities relatively more attractive places to induce faculty members to remain and new graduates to choose in place of industry. This will in turn accelerate the production of new researchers. At the best, we are still likely to have a serious shortage of NL researchers for the foreseeable future: given that there are only about 15 universities with serious graduate NL programs, with an average of 1.5 faculty members who each graduate one Ph.D. student per year, we can expect for the near future only 20 or so new NL Ph.D.s per year. Probably fewer than half of these will go to universities, and it will be five years before the new faculty produce their first Ph.D. students. Thus the NL manpower situation for the next ten years is likely, in the absence of any massive intervention, to show little improvement.

As mentioned above, a possible way to increase our NL capability in the shorter term would be to encourage a cross-over of faculty from areas such as linguistics or developmental psychology to AI NL research. There are already incentives for researchers with suitable backgrounds, since funding in linguistics has generally been cut along with social sciences. An infusion of researchers from these areas may have possible long-term benefits for AI NL research above and beyond providing more manpower; it is our belief that in order to truly succeed in producing general, robust NL systems, we must develop a far deeper science of human language understanding and language development.

It might also be helpful to provide graduate fellowships, though the most serious shortage seems to supervisors of research, not interested students or money to support the students.

3. *Expert Systems*

Expert systems have recently received a great deal of attention from the press, and have probably been responsible for putting to rest, one hopes forever, questions about whether AI could ever produce something useful. There are however serious questions about whether expert systems as they are now understood are capable of carrying the weight of paying for the rest of AI for very long, and if not, what research directions should be explored to encourage the continued transfer of useful products from mainline AI through the expert systems channel.

3.1. *An Assessment of the State-of-the-art in Expert Systems*

3.1.1. *The Players*

3.1.1.1. *Universities in Expert Systems*

Stanford University (Ed Feigenbaum, Doug Lenat, Bruce Buchanan, Mike Genesreth, ...) The Heuristic Programming Project is probably the best-known group in expert systems, comprising research and development of expert systems for problems in medicinal, circuit design, general expert system utilities, knowledge acquisition, automatic design of experiments, interpretation of mass spectrogram data, and other areas.

Rutgers University (Saul Amarel, Tom Mitchell, Sridrahan, C. Kulikowski) Research especially on the the learning and inference aspects of expert systems, on circuit analysis, and other systems.

Carnegie-Mellon University (John McDermott, Raj Reddy, Mark Fox, ...) Research into production systems as well as general architectures for expert systems, along with special problems in factory management, speech understanding, computer system configuration, and other areas.

MIT (Randy Davis, Peter Szolovitz, Jerry Sussman, Joel

Moses ...) Concentration on mathematical aids, automatic VLSI circuit design, circuit analysis, medical information systems, and other areas.

University of Pittsburgh (Harry Pople, ...) One of the original sites for research on medical diagnosis.

University of Illinois (Ryszard Michalski, Donald Michie, R.T. Chien) Research on general models of inductive inference and automatic rule formation, as well as special applications to medicine, plant pathology, tax planning, air traffic control, and other areas.

Ohio State University (B. Chandracakaran, S. Mittal) Research on medical expert systems.

University of Pennsylvania (Bonnie Webber, Tim Finin, Aravind Joshi, ...) Concentration on Interactions between users and expert systems.

3.1.1.2.*Companies in Expert Systems*

Teknowledge (Rick Hayes-Roth, Lee Erman, Stanford researchers) Closely associated with the Stanford Heuristic Programming Project, offers courses on setting up expert systems.

SRI International (Nils Nilsson, Rene Reboh, Bob Moore, ...) Commonsense reasoning, geological mineral exploration expert system, other areas.

Schlumberger-Doll Research Laboratory (Dave Barstow, ...) Automatic programming and programming assistants, interpretation of oil well logs.

Fairchild Advanced Research Laboratory (Peter Hart, Dick Duda, ...) Seismic data analysis system, VLSI circuit design aids, other areas.

DEC (John Ulrich, Hal Shubin) Computer system diagnostic tools.

Xerox PARC (Richard Fikes, Frederich Tou, Austin Henderson, ...) Intelligent database assistant system.

Hewlett-Packard Research Labs (Ira Goldstein)

Rand Corporation (Philip Klahr, ...) Air battle simulator.

EG&G Idaho (William Nelson) Expert system for diagnosis and treatment of nuclear reactor accidents.

IBM Thomas J. Watson Research Lab (Se June Hong, ...) Operating system expert.

Mitre Corp. Air traffic control expert systems.

3.1.2. *Where Expert Systems Stand Now*

For the purposes of this report, we will consider expert systems to fall into two major categories. (1) programs of the types that are now under development or in use, especially those which are based on production systems (sets of if-then rules); (2) any other programs designed to capture the knowledge or mimic the performance of an expert, especially those that use a variety of knowledge representation forms.

Most of the attention in the press has been given to systems of the first category. The ideas on which these systems are built go back at least 15 years to the MACSYMA mathematical aid system [Bogen et al 1974] and the DENDRAL system for finding chemical structure from mass spectrogram data [Buchanan et al 1969]. The heavily referenced, slightly more recent, MYCIN system for medical diagnosis [Shortliffe 1976] has inspired a large number of related systems, based on if-then rules. Most AI researchers would probably not have guessed even five years ago that this sort of production system model would ever be as successful as it has proved for medical and other expert systems. Other systems in the first category include R1, a system configuration expert for DEC VAX computers [McDermott 1980, 1981], and PUFF, a program for diagnosing lung diseases [Kunz et al 1978].

Despite their successes, such systems seem to be inherently limited in their ultimate applicability. Davis [1982] characterizes the current generation of expert systems as having (1) narrow domains of expertise; (2) fragile behavior at the boundaries; (3) limited knowledge representation languages; (4) limited input/output; (5) limited explanation capabilities; and (6) a single expert as knowledge base "czar".

Much of the current work on expert systems has concentrated on ways to keep the good features of the working systems, while improving the systems through evolution. Some of the good features include: (1) separation of the inference engine and the knowledge base; (2) use of as uniform a representation as possible; (3) keeping the inference engine simple; and (4) exploiting redundancy (e.g. overlapping rules) to make systems as non-fragile and complete in their coverage as possible [Davis 1982].

Thus a great deal of effort has been put into the problem of *knowledge acquisition*, that is, into methods for getting the knowledge that an expert uses into rules of the form that a current generation system can use. For example, EMYCIN (for "Empty MYCIN", the MYCIN medical consultation system with the domain-dependent knowledge and rules removed [Van Melle 1980]) has been used as a basis for constructing other expert systems, such as a personal tax expert [Michaelsen 1982] can use. Other work has been devoted to the identification and encoding of *metarules*, that is, rules about rules, that can help guide a system in its search of its knowledge base, and in the application of context-dependent knowledge and rules.

The second category of expert system, comprising much current research, is rather difficult to characterize, especially since many application projects across the entire spectrum of AI have adopted the fashionable term "expert system". We will reserve the term here for systems that attempt to model expert performance, or which could lead to such systems in a fairly direct way. Recent topics of interest have included the "commonsense algorithm" [Rieger 1975] and "qualitative process theory" [Forbus 1982] (models of common sense physics that can be used in a system that can reason about mechanisms such as steam systems or nuclear power plants or aircraft subsystems); mixed knowledge bases as in PROSPECTOR [Gashnig 1980], which contains three-dimensional geological knowledge about rock formations, soil composition, etc. that is not neatly capturable in a production system; automatic programming systems, systems that can accept specifications for programs and produce code, or that can help verify the correctness of code; VLSI circuit design and layout experts, and closely related design checkers; and others.

3.2. *An Enumeration of Problem Areas in Expert Systems*

There are problems for which the current generation of systems seems totally inappropriate. Some examples of types of problem domains which cannot be neatly captured

by systems of the sort in the first category include:

(1) problems that require a variety of representation types, that can not be easily operated on by a simple inference engine, and that may be difficult to combine in a single knowledge base; an example is the HEARSAY II system, which had a knowledge base with frequency spectrum, phonemic, syntactic, and other types of knowledge, all very different;

(2) problems that require procedural knowledge; for example a computer system troubleshooter [Davis 1982] that ought to apply general knowledge of system dependencies, interconnections, and operational knowledge for various components. Such knowledge, if coded in if-then rules, would require a massive data base of very similar rules, for example, listing every problem condition separately, rather than stating a set of general rules for detecting and correcting problem conditions;

(3) problems where a number of if-then rules will ordinarily be fired in a sequence, rather than being more or less independent. In such a case, it would be very inefficient to search through the entire rule base at each cycle.

(4) problems where the reasoning is based on *casual models*; current systems generally do not differentiate between conclusions that follow of necessity in a given situation, and conclusions that follow with a high degree of likelihood in a situation.

Given these inadequacies, there are serious problems in knowledge representation that must be solved, and furthermore, the knowledge representation problems must be solved with an eye toward eventual interaction with appropriate inference engines. Which task, the knowledge representation or the inference engine, is given priority, is another chicken-or-egg type problem.

Many of the problems enumerated for NL systems are also problems here, with different twists. It is difficult for universities to identify problems that are of manageable size for a single researcher, yet large enough to give a reasonable test to novel ideas. There is debate about the intellectual contribution of applying an existing system design to yet another problem domain. There are serious problems in identifying the nature of the knowledge needed to master a domain, and serious problems in systematically gathering all of the knowledge, and again there is a chicken-or-egg dilemma: should the knowledge gathering or knowledge characterizing be done first? How can we know that all the necessary knowledge has been gathered or taken into account?

Questions have also been raised about what the role of expert systems should be relative to humans. For example, should such systems be designed as decision-making aids for people, or as replacements for people, or as independent providers of second opinions? It is not wise to design a system for which there is no market: for example, Rodewall [1982] has pointed out that physicians rarely have a list of test results available in a form that could be easily fed to an expert system such as MYCIN. Instead, they generally take and keep information in a narrative form, which includes the time course of the patient's problems along with subjective reports of experiences, all of which may be extremely helpful (or perhaps sufficient) for the physician to make a diagnosis. This is also the form in which new cases and results are presented in medical journals. In order to make regular use of a MYCIN-like system, a physician would have to change to a different mode of operation from the one he or she was trained to use.

Other problems concern keeping knowledge bases consistent, and updating knowledge bases if items are found to be in error; automatic collection of expertise by "experience" or "observation"; explaining the reasons for a decision to a human user, so that the user can be convinced of the correctness of the decision; and many, many others.

3.3. *An Enumeration of Opportunities in Expert Systems*

We now have an existing technology which can be applied to a range of problem areas. The main difficulties are in (1) identifying areas which can be couched in terms of currently available systems; (2) gathering and representing all the relevant knowledge in the area (knowledge engineering); and (3) designing a system in the area which meets a need. This process, which lies somewhere between development and basic research *in the field to be represented*, should be repeated many times in the next few years, to build an epistemology of knowledge types, and to provide a variety of useful applications.

At the same time, research must be encouraged that can eventually streamline the process of knowledge base gathering and building.

New knowledge representation schemes must be developed, and methods developed for constructing expert systems that use a variety of knowledge representation schemes.

We also must develop ways of making appropriate decisions in the face of incomplete and varied knowledge bases. Especially important are ways of rapidly making plausible hypotheses if there is not adequate time for fully exploring a knowledge base, especially for real-time decision systems, such as on-board expert aircraft failure detectors and correctors.

3.4 *Recommendations for the Next 5-10 Years in Expert Systems*

I could use help here.

3.5. *An Assessment of the Resources Required in Expert Systems*

I could use help here.

4. *AI Support Software*

Good support software can dramatically improve the rate of progress, and the quality and portability of the research produced. The main kinds of support software of interest here are (1) programming languages for AI, (2) knowledge representation systems, and (3) programming environments. (In a certain sense, VLSI design layout and fabrication technology might be included here also, but the application of such support has no special relationship to applied AI as opposed to other fields. AI research in VLSI is being considered by other groups.)

4.1. *Programming Languages*

LISP has been the AI language of choice for a long time. The main variants in use now are INTERLISP (associated with Xerox and BBN and used on their LISP machines, and also used by SRI, USC/ISI, Stanford HPP, and other), FranzLisp (supported by Berkeley and CMU, and in wide use on VAXes), ZetaLisp (used at MIT and on

Symbolics and LMI Inc. LISP machines), MACLISP (used at MIT, Stanford, and elsewhere), T (Yale), NIL (MIT), LOGLISP (Rochester), as well as other LISPs with marginal use. In addition, efforts are underway at CMU, Utah, and other sites on ComLisp (for Common LISP), which its writers hope will become a standard for non-INTERLISP users. In addition, there are a variety of packages that can be added to LISPs to enhance their capabilities. These include packages for object-oriented programming such as FLAVORS (MIT), LOOPS (Xerox PARC), and QLISP (Stanford), and knowledge representation languages, which are treated separately below.

In recent years, PROLOG has become the AI language of choice in Europe, and has attracted a following in the US as well. Some AI people are interested in object-oriented languages such as SMALLTALK, as well as in other widely used languages, such as PL/1, PASCAL, APL, and FORTRAN (which has fairly wide use in computer vision, along with C, BLISS, assembly language and others).

The three main factors in the choice of language seem to be (1) familiarity -- people want to continue to use a language they know well; (2) availability -- not all languages are available on every machine, and some (such as the LISP machines) are constrained to a single language; and (3) appropriateness to the problem at hand, including ease of use. This last point is important. Some languages, such as INTERLISP or ZETALISP, are really total environments rather than just programming languages. They contain editors, compilers, screen managers, file managers, and elaborate debugging aids, all of which are available without ever leaving the programming language. Packages that integrate editors with LISPs, such as the EMACS/FranzLisp combination, are also popular, and have many of the same advantages as the previous systems.

4.2. *Knowledge Representation System*

Knowledge representation systems are often embedded in a LISP. They commonly support the storage of items in a hierarchy, with properties of an item in the hierarchy automatically inherited by all the daughters of the item. Many are based on the "frames" ideas of Minsky [1975]. Some of the KR systems that have been used at locations other than the place which designed them are:

(1) KL-ONE (BBN)
(2) FRL (MIT)
(3) UNITS (Stanford)
(4) SNePS (SUNY Buffalo).

Some KR systems that are either theoretically oriented, or which have not been widely distributed have also been influential, and may eventually lead to practical KR systems; these include:

(1) KRL (Xerox PARC)
(2) RLL (Stanford)
(3) First order predicate calculus
(4) Partitioned semantic networks (SRI).

APPENDIX B

CURRENT RESEARCH AND CRITICAL ISSUES IN DISTRIBUTED SOFTWARE SYSTEMS

by

John A. Stankovic
Krithi Ramaritham
Walter H. Kohler

University of Massachusetts
Amherst, Massachusetts

1.0 *INTRODUCTION*

A distributed software system is one that is executed on an architecture in which several processors are connected via a communication network. The main issue in the design of such systems is: how are programs, data and control to be distributed among the components of a distributed system? In this paper we consider this issue from the perspective of the distributed operating system, programming language and database areas. Section 2 summarizes current research in distributed operating systems and programming language areas. These two areas are discussed together due to their symbiotic nature. Section 3 discusses distributed database research. Critical issues in the design, implementation and evaluation of distributed software systems are itemized in Section 4. An extensive bibliography is also provided.

Given that distributed systems have been under investigation for a number of years now, a survey of issues in this area could have a very wide scope. We focus only on the issues connected with those software aspects mentioned above. Thus, our presentation here does not include such areas as the communication subnet, algorithms for distributed computing, hardware and architectures for distributed systems, and impact of other areas such as artificial intelligence on distributed systems.

2.0 *DISTRIBUTED SYSTEMS SOFTWARE*

Distributed systems software is usually designed and implemented in multiple levels. Each level provides the next higher level with an abstract distributed machine consisting of resources and primitives for using them. This section describes distributed systems software issues and current research for the operating system and programming language levels as well as their interaction.

2.1 *Background*

2.1.1 *Operating Systems* - Due to the wide variety of distributed systems there is a wide variety of operating systems controlling them. For purposes of discussion we simplistically divide these operating systems into three classes - network operating systems (NOS), distributed operating systems (DOS), and distributed processing operating systems (DPOS).

Consider the situation where each of the hosts of a computer network has a local operating system that is independent of the network. The sum total of all the operating system software added to each of these network hosts in order to communicate and share network resources is called a *network operating system*. The added software often includes modifications to the local operating system. In any case, each host can still act independently of the network and various degrees of sharing are possible. The most famous example of such a network is ARPANET and it contains several NOS's, e.g., RSEXEC [FORS78] and NSW 4[FCRS78]. Such operating systems are characterized by their having been built on top of existing operating systems.

Next consider networks where there is logically only one native operating system for all the distributed components. This is called a *distributed operating system*. The confusing element is that this one OS may be implemented in a variety ways. The most common is to replicate the entire OS at each host of the distributed system. One might be tempted to then consider this replicated piece as a local OS. In fact it is not because resources are allocated in a more global fashion, there is no exclusive, local administrative control, and, in general, there is no dichotomy from the user's point of view between being on the network or not. If we draw an analogy to the NOS, we see that the local replicated piece of the DOS is similar to a piece of an NOS with a null local OS. On the other hand, there is no requirement for a DOS to be implemented by replication and many designs are possible. For example, see Medusa [OUST80] and Micros [WITT80]. Distributed systems with DOSs are designed and implemented with network requirements in mind from the beginning.

Finally, a *distributed processing operating system* is a DOS with the added requirement that the OS must be implemented with no central data or control at any level. This is such a demanding requirement that there are probably no systems that meet these requirements if, in fact, they can be met. If such systems were possible, significant advantages are hypothesized [ENSL78, JENS78, STAN79]. If such systems are not possible, those approximating a DPOS would also provide similar advantages but to a lesser degree.

In Table 1 we attempt to categorize some of the current distributed computing OS research. We apolgize for those systems omitted. Note that all the OS's in the DPOS category are probably DOS's. However, those DOS's that seem to treat the special requirements of DPOS's are placed in the DPOS category. For additional information we list four other categories - distributed file systems' OS control languages, mail transport systems, and those object-based systems which are not already included in the other categories. The category "object-based' listed in Table 1 includes systems that are either centralized or deal primarily with the programming language - operating system interface. Research in the area of workstations, e.g. [BASK80] is not treated in this paper

TABLE 1: CATEGORIZATION OF OPERATING SYSTEMS FOR DISTRIBUTED COMPUTING

NOS	*DOS*	*DPOS*
NSW [FORS78]	Accent [RASH81]	ADCOS [STAN82]
RSEXEC [FORS78]	Apollo [APOL81]	Archons
XNOS [KIMB78]	DCS [FARB73]	CHORUS [GUIL82]

Domain Structure	Fully DF System
[CASE77]	[ENSL80]
Eden [LAZ081]	HXDP [JENS78]
Locus [POPE81]	Medusa [OUST80]
RIG [BALL76]	Micros [WITT80]
ROSCOE (Arachne) [SOLO78]	StarOS [JONE80]
TRIX [WARD80]	
UNIX (4)	
WEB [HAMI78]	

File Servers	*Object-Based*	*OS Control Lang.*
Cambridge FS [DIO80]	Argus [LISK81]	PCL [LESS80]
Felix FS [FRID81]	CAP [WILK79]	
Violet [GIFF79]	Hydra [WULF74]	
DFS [STUR80]	iMax [KAHN81]	

Mail Transport Systems

Smalltalk [GOLD80]
Grapevine [BIRR81]

The (4) after UNIX indicates that there are at least 4 extensions to UNIX for distributed systems including LOCUS and those extensions done at Bell Labs, Berkeley and Purdue.

2.1.1 *Programming Languages* - Languages for distributed systems, in general, contain message-based mechanisms for process communication and synchronization. Though most extant distributed systems have been programmed using *ad hoc* modifications to sequential languages, concurrent languages based on shared variables for process interaction are becoming more widely available. Such languages provide high-level constructs for the specification of processes and the interaction between processes. Languages such as Concurrent Pascal [BRIN75], Modula [WIRT77], and Mesa [MITC79] belong to this category. In these, process interaction is via resource sharing controlled by mechanisms based on the monitor concept [HOAR74].

Although the shared variable based approach is appropriate for processes in a network which interact relatively infrequently, it is not suitable for programming systems in which physically distributed processes cooperate and interact closely in order to perform a single task. For such systems, due to the presence of an underlying communication network, a message-based interaction is the most appropriate. Languages in this category are CSP [HOAR78], Gypsy [Good79], PLITS [FELD79] and SR [ANDR81].

There have also been attempts at designing languages in which facilities are provided for process interaction via both message-passing and resource sharing. This is essential in order to configure tightly-coupled processes as a single logical module that would execute on a single processor. Processes within a module would interact via resource sharing whereas Processes in different modules would interact via message-passing. These modules are called "guardians" in [LISK82], "resources" in [ANDR81] and "modules" in [COOK79]. These schemes are detailed in section 2.2.

2.1.3 *Issues*

The following are the main issues involved in the design of distributed systems software.

1 Structuring distributed systems,
2 Addressing distributed processes and resources,
3 Communication between distributed processes,
4 Nature of communication channels,
5 Management of resource sharing,
6 Decentralized control,
7 Protection mechanisms,
8 Error recovery, and
9 Distributed file systems.

Much of the distributed systems software research has been experimental work in which actual (prototype) systems are built. However, further work needs to be done in the evaluation of these systems in terms on the problem domains they are suited for, their performance, etc. This work is valuable because many research proposals are integrated into an actual system and it is important to know how well they coexist.

Of course there have also been investigations carried out independently of a particular system. This work is also treated in the discussions below. Overall, we feel that all the above issues are still unresolved in the context of NOS, DOS, DPOS, and programming languages for distributed systems and merit further work. In the remainder of this section we briefly treat each of the above research topics using examples from current research.

2.2 *Structuring Distributed Systems*

It seems attractive to structure a system based on the physical structure of the model associated with the problem. One of the advantages is that this utilizes the concurrency in the problem in a fairly natural manner. Programs written in functional languages and implemented on data-flow architectures are amenable to this structuring [COMP82]. Chang [CHAN79] has also described some graph algorithms using this approach.

The resource-server model in conjunction with object-orientation also appears to be conceptually attractive. There has been considerable work on both centralized [WULF74, WILK79, KAHN81] and decentralized [JONE78, POPE81, STAN82] object-based systems in the context of operating systems and programming language levels [LISK82, GOLD80]. Conceptually, an object is a collection of information and a set of operations

defined on that information. In practice, some systems treat the term object as simply a collection of information with common access characteristics. In either case, objects seem to provide a convenient and natural model for reducing the complexities in distributed operating systems. For example, moving a process to another processor after it has begun execution requires careful treatment of code, data and environment information. Treating each of these pieces of information as objects (or possibly the combination of them as objects) facilitates their movement.

Typically, associated with each object is a server which controls the execution of operations on the object and services requests for operations on the objects. Most systems are structured using the so-called resource-server model.

A node in a network is a collection of resources and processes, and is typically an autonomous entity. Processes outside a node access the processes and resources within a node via messages whereas intra-node communication is normally through shared memory (HXDP is an exception). Definition of the structure of a node requires the specification of a name for a node, the resources and processes that constitute the node and a set of interfaces for accessing these processes and resources. The structure of a node could change with time.

As mentioned above, programming languages have started addressing this issue, in particular, Argus [LISK82] (where the concept of a "guardian" is introduced), SR [ANDR81] (where the term "resources" is used) and *Mod [COOK79] (where "modules" are used for structuring systems). In Argus, a distributed program consists of a set guardians. Each guardian encapsulates a set of resources which are instances of abstract datatypes. Operations on these resources are controlled by the guardian which also provides schemes to handle concurrent access and system failure. These issues are discussed in detail in subsequent sections. Processes and data within a guardian exist and run on one physical node. In SR, a resource is defined along with a set of processes' called operations, to access the resource. In *Mod, a module is defined by the data structure definitions, procedures, processes and an external interface and is thus similar to a resource in SR.

Only by structuring real systems, will the adequacy of the resource-server model become clearer. Only Argus addresses the issue of failures of nodes by invoking concepts such as locking, atomicity, and checkpointing, concepts traditionally applied in databases. The problem of structuring distributed systems is complicated by the presence of truly distributed resources such as distributed databases, wherein resources need to be partitioned and replicated for achieving reliability and availability (see section 3). Node structuring schemes adopted in the above languages do not appear to be appropriate for distributed database applications.

2.3 *Addressing Distributed Processes and Resources*

The addressing issue [SALT78, SCHO78, DAVI81, BIRR81] in distributed systems is complicated because "names" are used in the context of so many different functions and levels. For example, names are used for referencing, locating, scheduling, allocating and deallocating, error control, synchronization, sharing, and in hierarchies of names - to name a few. Names actually identify a resource in a logical way, then are mapped to an address which in distributed systems is then mapped to a route. Names used at different levels referring to the same object must also be bound together. Allowing relocation and sharing of objects causes additional problems. Ideally, names should be bound to addresses and routes dynamically for increased flexibility, but this causes a potentially large and excessive run-time penalty. Broadcasting is often involved in implementing dynamic binding of names, addresses and routes. Heterogeneous distributed systems cause additional problems (more mappings required).

Addressing can also be categorized as implicit, explicit, path-based, or functional. These methods will be described below.

The simplest form of connection establishment is *implicit addressing*, where a process can only communicate with one other process in the system. This is usually the base for processes created to perform a single service and only communicate with their parent process. Though such a model is appealing in its simplicity, it is not flexible enough by itself to allow any pair of processes to communicate, a minimal requirement of a general communication facility.

Explicit addressing is characterized by the explicit naming of the process with which communication is desired [HOAR78, BRIT80]. This addressing scheme is particularly suited for process configurations in which the output of one process serves as input to another. Explicit addressing requires a global knowledge source containing the identity of each process in the system. Some systems have predefined names for processes providing system services and a user accessible table of user process ID's. A communication mechanism dependent on explicit addressing is an adequate basis for a complete communication system, as evidenced by its use in the Thoth system [CHER79]. However, explicit addressing by itself is not flexible enough to effectively handle such common circumstances as process migration in distributed systems and multiple processes providing a single service.

Path-based addressing associates global names with message receptacles, or 'mailboxes'. A process can declare a mailbox into which messages can be received and can specify a destination mailbox when sending a message. Such a scheme is used in [GOOD79] wherein the mailbox is global to the processes that use it. Recently, use of a distributed but globally accessible memory for process communication is described in [CELE82].

Functional addressing establishes a connection based on the need to serve or request a service. In this case the communication path is itself a named entity of the system. For the user of a path, the identity of the process or processes on the other end of the path is insignificant. What is significant is that they are providing a service or that they are requesting that a service be provided. This scheme is flexible because an individual process is not necessarily associated with a communication path, and paths themselves may be passed within messages. This kind of addressing was introduced in [BALZ71] and expanded in [WALD72]. Functional addressing is the underlying concept used in many current languages, most notably in the following: PLITS [FELD79] and *Mod [COOK79] where "port" denotes a path, Distributed Processes [BRIN78] and Ada [ADA80] where "entry" denotes a Path, and SR [ANDR81] where "operation" denotes a path. It is also used in the following message-based operating systems:

Accent [RASH81] where 'port' denotes a path, and DEMOS [BASK77], where 'link' denotes a path.

Addressing in most languages is static in that the processes accessible to each process is fixed at compile time. However, some languages, for example, Gypsy [GOOD79] and PLITS [FELD79], do support dynamic addressing whereby it is possible to interact with a changing environment.

It should be clear that the various forms of addressing described above need different kinds of operating system support. The complexity involved in providing this support and its effect on system efficiency are questions that remain to be answered. Open problems in this area include, what type of name is required at various levels in the distributed system, making the mappings flexible and efficient, how to deal with heterogeneity, supporting relocation of objects, supporting multiple copies, and how to perform process naming when processes can move during execution.

2.4 *Communication Between Distributed Processes*

There are three aspects of communication between processes: primitives used for message-passing, techniques used for buffering messages, and the structure of messages.

An essential issue in the discussion of *communication primitives*, particularly in a distributed environment, is the degree to which they provide concurrency between communicating processes. Communication primitives use process blocking to achieve synchronization. (See [LISK79]) for extended discussions on communication primitives.)

A procedure call can be simulated in a message oriented approach to process communication with the *reply-send* primitive (also referred to as the remote invocation send [LISK79]). (Consult [NELS81] and [STAN81] for an extended comparison of message-based and remote procedure call approaches to communication.) This primitive is commonly used for programming resource-server interactions and hence languages provide higher-level language support for this type of interaction. The remote procedure call can be implemented either by having a single cyclic process, as in *Mod [COOK79] and SR [ANDR81] for each type of call, by creating a new process for each execution, as in Distributed Processes [BRIN78] and Argus [LISK82], or by programming the server process with separate receives for distinct calls, as in Ada [ADA80]. With the *synchronized send*, the sender is blocked until the message is received by the destination process [HOAR78].

The reply-send and synchronized send-primitives require no message buffering or queueing if only a single process has send access to the path.

The *no-wait* send (or asynchronous send as discussed in [GENT81]) maximizes concurrency between communicating processes. In this model, the sender resumes execution as soon as the message is composed and buffered within the communication facility. The no-wait send is the most flexible of the three send primitives, though it introduces extensive implementation problems. The need for buffering introduces problems fo flow and congestion control, addressed in the following section. Additionally, the sender is not notified of transmission errors or communication failure.

There are two forms of receives: unconditional and conditional. The *unconditional receive*, or blocked receive, blocks the receiver until a message is queued on the selected port. The unconditional receive sacrifices process concurrency by blocking the receiver when no messages are queued. It has two variations. Frequently it is desirable to block on a set of paths. The first message received on any of the specified set of paths is returned to the receiver. This is useful for a serving process awaiting messages from multiple sources. A second, more important variation is the *blind unconditional receive*. This form of unconditional receive blocks on all paths to which the requesting process has receive access [STEM82].

The *conditional receive* (or selective receive as discussed in [RAO80]) introduces a polling capability, allowing the receiving process control over the degree of concurrency desired. The conditional receive polls a (set of) port(s) for a queued message. If a message is present it is returned, otherwise control is returned with a flag indicating no message is available. The conditional receive can be generalized to check for a more complex condition than the presence or absence of a message.

Implementing a communication facilty requires the ability to *buffer messages*. Regardless of the buffering technique used, buffer maintenance introduces the problem of resource allocation. There are three basic techniques, port-local allocation, process-local allocation, and system-global allocation, and a hybrid called port-global allocation.

Port-local allocation, assigns a fixed buffer space to each port [RASH81]. When the buffer space is filled and a process attempts to send another mesaage to the port, there are three alternatives. First, if a flow control option is included in the send primitive, the port can dynamically expand its size to allow an additional message. Second, the sending process can remain blocked until space is available on the port for an additional message. Third, an error flag can be returned to the sending process indicating a full port.

The *process-local allocation* technique associates the buffer space with the process rather than the port [KNOT75]. This restricts the total number of outstanding messages for a process.

A third resource allocation alternative is to maintain a global buffer pool (examples include Rosoce [SOLO79] and UNIX [RITC74]). This technique, termed *system-global allocation*, introduces a bottleneck in the system. A viable solution to the two problems mentioned above is to combine the port-local and system-global techniques. The *port-global* hybrid associates a fixed buffer space to each port while reserving buffer space with the system.

An important aspect of *message structure* is the typing of data within messages. Heterogeneous computer networks [ANDE71, LEVI77, BACH79] cause problems because of the different internal formatting schemes that exist. Normal solutions to the data incompatibility problem include defining a intermediate "standard" and providing a mapping to and from the intermediate standard. Further complications arise when there are precision loss and data type incompatibility.

Additional problems exist for internetwork mappings (gateways), including naming [SCHO78] and routing.

Some systems are purporting the benefits of strongly typed

data within messages (Eden [LAZO81], CLU [LISK78], and Accent [RASH81]). Strong typing of data provides reliability in general, but in a distributed system consisting of heterogenous nodes, it is especially important in ensuring proper conversion of messages across nodes [LISK82a].

Another issue under message structure is the length of the messages. The Roscoe [SOLO79], StarOS [JONE79], and Thoth [CHER79] systems have short, fixed-length messages. This is in part due to the belief that most messages in such tightly coupled systems are short control messages. Special, synchronous communication paths are provided for large data transfer, such as reading and writing files. Medusa [OUST80], Accent, and CLU [LISK79] provide for variable length messages, and, in general, most protocols for loosely coupled systems support variable length message transfers. Variable length messages are obviously more difficult to implement due to the buffering problems they induce.

2.5 *Nature Of Communication Channels*

The primary considerations in the design of communication channels in a distributed system are directionality, ownership, frequency of use, transference rights, and topology.

Directionality concerns whether or not a single process has both send and receive access to a single communication path. If so, the path is bidirectional (or duplex, as used in TRIX [WARD80]). Most systems provide only unidirectional (or simplex) paths, in which a process may send or receive messages over the path, but not both. In systems providing only unidirectional paths, bidirectional communication can be simulated via a pair of paths.

Ownership deals with the capability to destroy the path and terminate communication without consent of all processes with access to the path. Ownership can be associated with the directionality of communication, as in the Accent system where the allocator of a path automatically has ownership and receive access to the path.

Frequency of use deals with the limitations on the frequency a path can be used. The DEMOS [BASK77] and Roscoe [SOLO79] systems have a 'reply' path created for the sole purpose of returning a single message. Such a path is automatically destroyed after it is used once.

Transference rights concern the ability to duplicate a path, or pass access and/or ownership rights of an established path to another process. For example, a Process with ownership rights to a path can send that privilege to a receiving process, allowing it to destroy the path or change the path's characteristics. Transferring access rights can, but does not necessarily, imply loss of the transferred right.

Topology of a communication path is the set and structure communicating processes using the path. The topology can take four general forms: one-to-one, one-to-many, many-to-one, and many-to-many. Communication paths that are not one-to-one can often be functionally equivalent to a set of one-to-one paths, an important issue in examining topologically complex path structures. The Process Control Language (PCL) [LESS79] provides a complete specification of a wide variety of topological structures of unidirectional communication paths. The terms used below are introduced in the PCL description.

The *simple* path is provided for a one-to-one connection. A single queue is maintained to hold all messages. The *broadcast and multiple read* connections are one-to-many connections, associating a set of receivers with each sender. Every message sent on a broadcast communication path is received by every receiver; any message sent on a multiple read path is received by the first process requesting it. The broadcast path can be easily simulated by a set of simple paths from the sending process to the set of receivers. The multiple read path is useful when a set of servers are providing a service, and the service is performed equally well by any of the servers. A multiple-read path cannot be simulated with one-to-one paths without additional management control within the processes performing the service.

Many-to-one connections are provided in the *multiple-write* and *concentration* communication paths. A multiple-write mapping is the converse of the multiple-read mapping: every message sent by any sender is received by the receiver. This also can be simulated by forming a set of simple paths from the sender processes to the receiver. The concentration path can be used for synchronization of the set of senders: every message received is the concatenation of the set of messages from a single send by each sender. A message cannot be received until every sender has transmitted a message.

Many-to-many paths are combinations of one-to-many and many-to-one paths. The *multiple-write/broadcast* transmits every sent message to each receiver. A message transmitted on a *multiple-write/multiple-read* path by any sender is only received by the first receiver requesting it. A *concentration/broadcast* message is the concatenated messages from each sender and is sent to every receiver. A *concentration/multiple- read* message is the concatenated messages from each sender and is received by the first receiver requesting it.

The topology of a path is closely related to the transference and access rights of processes to the path, and may change over the lifetime of the path.

Current programming languages do not appear to have the facilities to specify the above restrictions on communication paths. However, a need for such facilities is apparent, especially for the designer of a protected system [STEM02] wherein protection of resources and processes is achieved by restricting the use of communication paths.

2.6 *Management of Resource Sharing*

While many resources must be shared in a distributed system, in this report we concentrate on scheduling research (sharing of processors). Most of the research on scheduling for distributed systems can be considered "task assignment" research and can be loosely classified as either graph theoretic [STON77, STON78a, STON78b, BOKH79] queueing theoretic [CHOW79, KLEI81, AGRA82], based on mathematical programming [CHU69, CHU80, MA80], or heuristic [GONZ77, ELDE80, BRYA81, CHOW82, EFE82, STAN83]. In all these cases a task is considered to be composed of multiple modules and the goal is to find an optimal (or in some cases a suboptimal) assignment policy for the modules of an individual task. Typical assumptions found in "task assignment" work are: processing costs are known for each module of the task, the interprocess communication (IPC) costs between every pair of modules is known, IPC cost is considered negligible for modules on the same host, and

reassignment does not occur.

Other scheduling research has been based on the bidding scheme [FARB73, SMIT80] where specific tasks are matched to processors based on the current ability of the processors to perform this work. These schemes are suboptimal, but are more extensible and adaptable than many of the other approaches. However, the cost of making and acquiring bids may become excessive, and the factors to use in making the bids have not been extensively studied.

Other research in scheduling on networks has been performed in the context of operating systems. We now discuss some representative examples.

The Distributed Computer System (DCS) [FARB73] was the first system to perform a form of dynamic load balancing using a bidding technique. This system used percentage of available memory as a load indicator. Incoming jobs are then routed to processors with the most memory available. Information concerning current system status is gathered by means of a broadcast message. However, no analysis, comparisons or measurements were performed to determine the effectiveness of this scheme.

StarOS [JONE78] is a message-based, object-oriented multiprocessor operating system. StarOS has been implemented on the Cm* multi-microprocessor. StarOS is one of two experimental operating systems for Cm* [SWAI77]. The other is Medusa and is described next. One main idea of StarOS is the task force, a large collection of concurrently executing processes that cooperate to accomplish a single task. The structure and composition of a task force varies dynamically and it is the unit for which major resource scheduling decisions are made. Even though StarOS is designed for a multi-microprocessor, we believe that the task force concept could also be used for more loosely coupled distributed systems. The scheduling function itself is divided between processes which are called schedulers, and a low-level mechanism called the multiplexor. The schedulers decide which environments are to be loaded and the multiplexor performs the actual loading. As far as we know, only very simple scheduling algorithms were implemented.

Medusa [OUST80] is an attempt to capitalize on the architectural features of Cm*. Medusa is implemented as a set of utilities (OS functions), each utility being a task force (task force has the same meaning as described above for StarOS). Each utility contains many concurrent, cooperating activities. Load balancing is done by automatically creating new activities within a task force to handle increased load and automatically deleting activities when the load is reduced, by coscheduling (an attempt to have different activities of the utility executing simultaneously on different hosts), and by pause time (a short time in which the context of a process remains loaded after an interrupt to determine if it will be reactivated). The pause time concept is supposed to reduce context swaps.

The "domain structure" operating system directly addresses the scheduling and movement of individual processes. It does this by describing the structure of domains necessary to facilitate movement of processes that are in execution. However, the scheduling algorithms are completely heuristic and contain weighting factors which are left unspecified, presumably to be tuned as system parameters. Interaction between processes and scheduling decisions taking these interactions into account is not discussed.

Micros [WITT80] has a unique scheduling algorithm called Wave scheduling. This algorithm is appropriate for extremely large networks. The Wave scheduling algorithm coschedules (assigns) groups related tasks onto available network nodes. The scheduling managers themselves are distributed over a logical control oligarchy and send waves of requests towards leaves of the control oligarchy attempting to find enough free processors. In fact more processors than required are requested because some parts of the control oligarchy may not be able to supply the necessary processors. This is a form of probabilistic scheduling.

Finally, there is a similarity of some of the above described scheduling research and research in routing algorithms. For examples, see [MCQU74, GALL77, SEGA77].

2.7 *Decentralized Control*

Depending on the application and requirements, decentralized control can take on various forms. Although research has been active for all types of decentralized control, the majority of the work is based on extensions to the centralized state-space model and can be more accurately described as decomposition techniques, rather than decentralized control [CHAN69, FU70, JARV75, CHAN75, AOKI78, HO80]. In such work large-scale problems are partitioned into smaller problems, each smaller problem being solved, for example, by mathematical programming techniques, and the separate solutions being combined via interaction variables. The interaction variables normally model very limited cooperation. See [LARS79] for an excellent summary of these types of decentralized control (decomposition).

A number of surveys relating to decentralized control have also appeared including [SAND78]. These surveys note the unclear meaning of optimality for decentralized control and hypothesize the need for a completely different approach. There has also been proposed a novel structure for decentralized control based on the concept of a "domule" which is a combination of a decision agent and its subsystem model [TENN81a,b]. Using this concept, interesting heuristics are proposed for decentralized control, but these are also largely based on decomposition.

Two forms of decentralized control are more appropriate to our current discussion: decentralized control that arises in distributed databases, and decentralized control that arises for stochastic replicated functions, such as routing and scheduling. By replicated functions we mean that the decentralized controllers implementing a function are involved in the entire problem, not just a subset of it. In distributed databases concurrency control algorithms are sometimes implemented as decentralized control algorithms. Solutions to this problem are known. However, in the most general form solutions to decentralized control of stochastic replicated functions are not known, e.g., Team theory [HO80] and various forms of control theory all fall short in dealing with this problem [STAN82]. Some preliminary work has been done in this area of decentralized control for stochastic replicated functions [LELA80, STAN83]. Possible approaches might include the use of random graphs, decision theory [RAIF61, LIND81], decision theory under uncertainty [HOLT71], stochastic learning automata [NARE74, GLOR80], and Bayesian decision theory [WINK72, STAN83].

2.8 *Protection Mechanisms*

The notion of capabilities, an often occurring notion in the area of protection, was first introduced into an operating system by Dennis and Van Horn [DENN66]. Systems employing a capability mechanism view all data as strongly-typed objects which are distinguished from one another by unique identifications. A capability consists of an object identifier and a set of access rights, which allow the manipulation of the object with a subset of the operations defined by the object's type. Capability-based protection mechanisms are difficult to implement. Once a process obtains a capability it is essential that it not be able to modify it. Capabilities may be stored as C-lists, as with the Intel 432 [COX81], or as tagged memory, as with the IBM System/38 [BERS80]. The C-list scheme stores all capabilities in separate capability segments. A major difficulty in the use of capabilities is the separation of data and capabilities. Tagged memory requires each unit of data, 32 bits in the case of the System/38, to be tagged to indicate whether it is a capability or not. Tagging presents a large overhead for memory. See [LEVY81] for a complete description of capability-based architectures.

Object-oriented systems such as CAL, Argus [LISK82], CAP [NEED78], Hydra [Wulf74], and the Intel 432 associate groups of objects with modules. A process within a module has the right to distribute access rights, in the form of capabilities, to processes outside the module. The protection of an object is controlled by the distribution of capabilities by processes within a module.

Encryption is one essential ingredient of a protection enforcement mechanism in distributed systems. It is used to authenticate messages exchanged via the communication medium [NEED78].

Protected distributed systems entail a certain amount of overhead in that each access needs to be checked. Schemes that minimize this overhead require further investigations.

But for a few exceptions [LISK82], current programming languages do not have facilities for specifying protection requirements. However, in object-oriented system, users should be able to specify requirements for protecting their own objects from each other as well as from other users.

2.9 *Error Recovery*

Exceptions in distributed systems can occur due to data transmission errors and process control errors. Data transmission errors, including lost messages, the receipt of garbled messages, duplicates, and misdirected messages should be transparent at the process level. It is the responsibility of the communication facility and its underlying protocols to ensure reliable, error free data transmission across communication paths between processes. This requirement results in all remote procedure calls having an exactly once semantics whereby a call terminates after the called procedure has been executed exactly once in spite of system failures.

Process control errors take the form of processes involved in a deadlock or a livelock, and destroyed processes (including node failure). The deadlock problem has been studied in various contexts. See [ISLO80] for an excellent survey of deadlock in centralized systems. Additional work has been published for distributed systems but it is largely concerned with resources in distributed databases.

Little has been written about deadlock in the context of NOS, DOS and DPOS's. One exception is the Medusa system where deadlook avoidance is used. Here, functions provided by the utilities (OS functions) are divided into service classes such that (i) a single utility provides all the services in each class, and (ii) there are no circularities in the dependencies between classes. Furthermore, each utility must contain separate pools of resources so that it can provide independent service to each class. These conditions avoid the deadlock problem. More work is required for all approaches to deadlock in distributed systems.

Now we examine specific cases of deadlock. The blocked receiver problem occurs when a receiver blocks on an unconditional receive and no message is ever delivered to the process. Unexpected process destruction is the most likely candidate for causing a blocked receiver problem. The IPC facility must take responsibility for notifying other processes attached to the destroyed process. The IPC facility should notify the receivers connected to the communication paths to which the destroyed process has send access. Processes with send access to a port connected to a destroyed receiver should be notified on the next attempt to send a message.

The deadlock problem is a direct result of blocked or destroyed processes preventing further communication. If blocked receivers time-out and senders are properly notified, deadlock cannot occur. If either of the features is not included in the communication facility, it is necessary to actively detect deadlock.

Programming language designers have only recently started paying attention to the specification of actions to be taken in the event of a failure. An often adopted solution to exception handling is a time-out, in which the processes indicate a specified time limit for waiting. A sending process specifies the time within which it expects its message to be received whereas a receiving process specifies the time within which it expects the next message to arrive. When the time limit is exceeded, control returns to the process concerned with a message indicating the time expiration [ADA80, LISK82]. In PLITS [FELD78], the notion of transaction keys can be used to notify processes about errors. For instance, a process can be programmed to wait for a message by transaction key only, without specifying the source of the message. Thus, failure of a sending process can be intimated to the receiving process by using the appropriate transaction key.

Also, in Argus [LISK82] the notion of atomic action is used to cope with errors. An atomic action appears to a user to have been executed in exclusion and thus gives users the facility to program remote procedure calls with the exactly once semantics. Their implementation requires facilities for looking and unlooking data items such as those available in databases. Recovery and roll-back to some previous system state may be another option. In [RUSS80] a set of primitives are proposed for state restoration in distributed systems.

An entirely different approach, still in its infancy, to dealing with errors in distributed systems is to construct probably-correct systems from the start. Gypsy [GOOD79] facilitates the development of formally correct programs along with exception-handling mechanisms. Proofs make use of the history of message passing in the system. Several proof methods have been proposed for CSP programs [APT81, CHAN81]. Proofs for the absence of deadlock in distributed systems can be found in [CHAN79]. Proofs of systems structured using the

resource-server model are presented in [RAMA82]. The issue specifying and proving-network protocols have also been receiving attention. Some of the approaches are based on history variables associated with message passing [GOOD79] or on state machine models [BOCH78]. Recently techniques based on temporal logic [MELL81] have also been investigated. These attempts have to be further developed before they can become practicable for constructing correct distributed systems.

2.10 *Distributed File Systems*

Research in distributed file systems can be considered as providing an important but only incremental step in establishing integrated distributed computing systems. Three representative distributed file systems are now briefly described.

The Xerox Distributed File System (DFS) [STUR80] is used as a basis for database research. The DFS is based on a server model where multiple servers may cooperate to service a single transaction in an atomic fashion. Multiple files may be involved in a single atomic transaction. To provide high concurrency, DFS provides fine-grain looking at the byte level. Access control is provided by an authentication server that is based on capabilities. File replication is not supported. Users access the DFS through programs called clients that run on the user sites The DFS approach to distributed file systems is sometimes called the client-server model. Overall, DFS gives the illusion of a single, logical file system.

The Felix file server [FRID81] is designed to support a variety file systems, virtual memory and database applications. It is the only storage component in the system. An atomic transaction in Felix may involve single files or a set of files. A highly flexible mechanism for file sharing is supported by using six access modes (read copy, write copy, read original, write original, read exclusive and write exclusive). The level of locking granularity is the block level. Access control is based on capabilities and no file replication is supported. Felix is also based on the client-server model.

LOCUS [POPE81] is a distributed file system that is application code compatible with UNIX [RITC74]. In contrast to the above two systems LOCUS does support file replication. A centralized synchronization mechanism is used to maintain mutual consistency among replicated files as well as to synchronize multiple accesses to shared files. LOCUS continues to operate even though there is partial system failure or network partitioning. A concept based on version vectors is used to resolve conflicts at system recovery time. LOCUS is not designed as a client-server model but as a file system integrated with the rest of the operating system.

Many of the open issues for distributed file systems are the same as for distributed databases (see Section 3). These include how to maintain the atomic property in the presence of crashes and other failures, support of multiple copies, concurrency control and deadlock resolution. Other issues include directory assignment, replication and partitioning, where to divide the responsibility between file servers and clients, and whether or not to embed the distributed file system in the operating system.

3.0 *DISTRIBUTED DATABASE MANAGEMENT*

In this section we give a brief survey of the current research associated with the development of general purpose distributed database management systems (DDBMS's). Rothnie and Goodman [ROTH77] state that "distributed database management is an attractive approach ... because it permits the database system to act *conceptually* as a centralized system, while *physically* mirroring the geographic distribution organizations..." Some of the potential advantages of distributed database management systems are: easy access to geographically distributed but logically integrated data from a single site, increased reliability and availability, faster data access, and incremental system growth.

3.1 *Background*

The *transaction concept* has emerged as an abstraction which allows programmers to group a sequence of actions into a logical execution unit. If executed atomically, a transaction transforms a current consistent state of the database into a new consistent state [ESWA761]. The virtues and limitations of the concept are described in [GRAY79]. It is the job of the transaction processing component of a DDBMS to preserve atomicity of transactions. In order to do this, protocols for resolving data access conflicts between transactions (concurrency control protocols) and protocols for recovering a consistent state of the database in spite of user errors, application errors, or partial system failure (processes, nodes, links, etc.) are necessary [KOHL81]. There is general agreement that the *logical structure* (system architecture) of a distributed database management system can be described by Figure 1. The four basic components are transactions, transaction managers (TM's), data managers (DM's), and data. A description of this model can be found in [BERN81a].

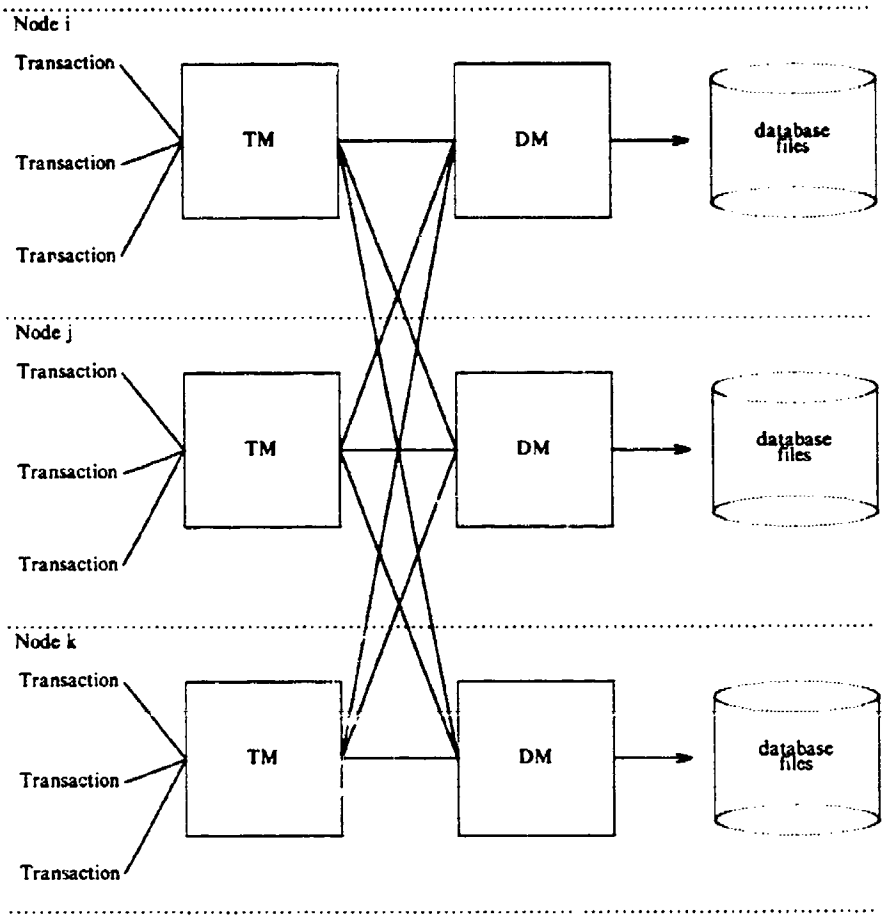


Figure 1. DDBMS System Structure

3.2 *Synchronization (Concurrency Control)*

Bernstein and Goodman [BERN81a, BERN82a, BERN82b] have developed a unified framework, called *serializability theory*, for analyzing the correctness of concurrency control algorithms and have shown that most of the commonly known methods can be understood using a few basic concepts: read-set and write-set, schedule, serial schedule, legal schedule, conflicts, dependence relation, read-write conflict, write-write conflict, serializable schedule, and equivalence of schedules. See the listed references for definitions of these terms.

The theory partitions the synchronization problem into two independent sub-problems: read-write and write-write synchronization. Each concurrency control scheme is a combination of policies to solve the sub-problems and each policy can be implemented using combinations of mechanisms. The three major classes of concurrency control policies are: locking 2-phase locking), timestamp ordering, and validation (also called the optimistic approach). There is also a variety algorithms to support each of these policies. Several hundred variations have been identified for the distributed environment [BERN81a, BERN82a] and we will make no attempt to list the very large number of research papers which have been dedicated to the study of this topic.

Any "new" methods for concurrency control are expected to be combinations and minor variations of known methods, so research aimed at discovering "new" methods is unlikely to be successful. However, even though many variations are known, little is known about how the choice of a scheme will affect system performance. Some simulation and queueing studies have compared performance, but the results are still inconclusive. In fact, since the models used in the studies have not been validated their results are limited. Several experimental studies involving testbed systems are either underway or planned. The results from these studies should be helpful, but it is unlikely that one approach will always be the "best" due to the wide variety of applications and system structures. These studies should provide enough evidence to determine if the choice of concurrency control algorithm has a primary or secondary impact on performance compared with other factors [BALT82].

The correctness of a concurrency control algorithm is judged by the serializability of all the schedules allowed in the scheme. That is, if all the legal schedules of a scheme are serializable, then the scheme is correct with respect to the serializability requirement.

Serializability represents the strongest degree consistency. Some applications may require only lower degrees of

consistency by allowing, for example, reads from multiple objects without concern for time consistency between the values read. Gray has defined degree 1, degree 2, and degree 3 consistency [GRA78], with degree 3 being the strongest. Systems that set read and write locks but release read locks before the end of the transaction are degree 2. They may have problems due to unrepeatable read. Systems that set no read locks at all are called degree 1.) Gray suggests that the performance penalty of degree 3 consistency is small and consequently lower degrees of consistency are a bad idea.

3.3 *Deadlock Resolution*

Within database systems, deadlock is a circular wait-for condition which may arise when a locking scheme is used for concurrency control. It may occur among a set of transactions as a result of the decision to wait for a lock because of lock mode incompatibility between the granted lock and the request. The circular wait-for condition prevents any transaction from proceeding unless one or more of them are chosen as victims to be aborted. Most of the concurrency control schemes using locking with delay incorporate a deadlock resolution mechanism, but a deadlock-free looking policy can be designed using deadlock avoidance or prevention mechanisms as well.

We classify the mechanisms into four categories. More discussion of these approaches can be found in [MENA79, OBER82, CHAN82, ROSE78, KORT82, ANDR82].

1 *Time-out.* One of the simplest ways to resolve deadlock is to specify a maximum wait time, and roll back the transaction which is waiting if the time expires before the request is granted.

2 *Deadlock Avoidance.* Deadlock can be avoided by requiring each transaction to preclaim all the objects to be accessed before execution. In this case, if a lock request cannot be granted, all requests are aborted before execution begins.

3 *Deadlock Prevention.* As an example of a prevention mechanism, we discuss briefly two protocols proposed by Rosenkrantz and Stearns [ROSE78] which use timestamps to define a priority. That is, an older transaction (one with a smaller timestamp) has higher priority than a younger transaction. The basic philosophy is that an older transaction is favored when conflicting with a younger transaction on the assumption that it may have already used more resources and thus more cost will be paid for rollback and restart.

o Wait/Die Protocol: If the requester (of a lock) is older (smaller timestamp), then it is allowed to *wait*: otherwise, it *dies* (die means rollback and restart).

o Wound/Wait Protocol: the requester is older, then the requester *wounds* the conflicting younger transaction, otherwise, the requester *waits*. "Wound" is an activity in which the requester sends messages to all the nodes the younger transaction has visited saying that it is wounded and the scheduler will abort a wounded transaction if the transaction has not initiated termination. If the younger transaction is terminating, i.e. in two-phase commit stage, then the wound message is ignored to save some unnecessary rollback/restart.

See [ROSE78, KOHL81] for more details.

4 *Deadlock Detection.* The detection mechanisms are mostly based on finding cycles in a transaction wait-for graph (TWFG). The nodes of the graph represent the waiting transactions, and the directed edges indicate for which transactions a given transaction is waiting. When cycles are detected, they are broken by choosing victims to be rolled back and restarted. This mechanism seems to have greater popularity than other mechanisms. However, in distributed systems a global transaction wait-for graph (GTWFG) must be constructed to detect deadlock involving two or more nodes. The GTWFG is maintained by exchanging messages among the nodes, causing additional comunnication overhead and delay. When a detection mechanism is chosen, some other decisions have to be made by the designer:

o Detector organization: centralized detector, distributed detectors, or something in between, e.g., hierarchically organized detectors with a detector for a cluster of nodes and a higher-level detector for a cluster of detectors.

o Detection initiation: is detection made periodically using a predefined time parameter, is initiated every time a transaction has to wait, or when any suspended transaction has waited for more than a predefined time period.

o Failure resiliency: establish backup for a centralized detector and activate when centralized detector fails.

The importance of an efficient detection algorithm for performance and which is the best approach are issues which have not been adequately resolved. It clearly depends on the average frequency of deadlock in the database application environment. A study by Gray [GRAY81] showed that the frequency of deadlock goes up with the square of multiprogramming level and the fourth power of the transaction size.

3.4 *Recovery (Failure Handling)*

It is the responsibility of the DDBMS to insure that transactions are *atomic* in spite of user errors, application errors, or partial system failures. This means that if a failure occurs to prevent the successful completion of a transaction, all database entities that the transaction modified must be restored to their state prior to the transaction. The same mechanism is used by the concurrency control scheme to roll back one or more transactions to resolve deadlock. This rollback is normally achieved by using a "shadow paging" mechanism or a "write ahead log" [TRAI82b]. Operating system support is needed to make these mechanisms more efficient.

Protocols for preserving transaction atomicity are called *commit protocols* [SKEE81]. They are used to insure the completion (commit) or rollback (abort) of the actions of a transaction at all sites. When multiple sites are involved, the question of site autonomy arises [LIND80b]. It is well known that when the common two-phase commit protocol is used, a site loses its autonomy to independently roll back once it enters the the second phase of the two-phase protocol. Skeen calls this a blocking protocol because a site failure under certain circumstances may leave operational sites blocked waiting for the failed site to recover. He has proposed an extension, called a three-phase commit protocol, which has the property that it is nonblocking if certain assumptions about the system behavior are true. The nonblocking property is achieved by using additional

messages. More research is needed to evaluate if the assumptions on system behavior are realistic and if the performance penalty of the additional messages required by the third phase are worth it.

While there has been some recent research on recovery protocols and rollback mechanisms, an adequate theory and formal model for recovery in a distributed system has not yet been formulated. More research is required to address this and related problems of failure detection, network partitioning, and node restart and integration [GARC82].

3.5 *Data Models Supported*

By restricting the choice of data model, there is more potential to optimize performance. The relational model has been the choice of researchers in DDBMS's. One reason is the opportunity for optimizing query processing by decomposing the query into subqueries which can be performed at remote sites [BERN81b]. The choice of a decomposition depends on the processing costs, the communication costs, and the data distribution. There continues to be some interest in developing optimal decompositions which minimize the processing and communication costs [CHU82]. However, even with a very simplified model of the distributed database and of the cost of processing a query, the costs of computing an optimal solution are large. Experimental work is needed to test the suitability of the model and the real savings before the utility of this research can be evaluated.

Due to the large number of existing DBMS's, a very practical problem is how to map between them or how to provide a single intermediate model which will enable an application to access heterogeneous systems. The research trend seems to be towards higher level non-procedural semantic data models [HAMM81]. There is also interest in integrating database support into the operating system.

3.6 *Data Replication and Location Transparency*

In order to enhance data availability and reliability, it is desirable to support the replication of critical data at multiple sites. Data replication can be implemented by defining two levels of abstraction [TRAI82a]: database entity --> database objects, and request --> actions. Requests and entities are the elements in the higher level abstraction. These are implemented by actions on one or more physical copies of the data. When data is replicated, it should be transparent to the end user to avoid problems of inconsistency which the user might introduce if updates were improperly done.

The high cost of replication in terms of increased delay and storage requirements makes the fully redundant case impractical in most situations [GARC79b, BARB81]. Many applications would prefer to have those parts of the database which have a high read-to-update ratio replicated at the sites from which access is frequent. For the majority of data, replication is not expected to be cost effective. Where it is justified, the number of redundant copies will probably be small (two or three copies). A simple partitioned database where the sites and network are reliable should be adequate for many applications.

Knowledge of the physical location of the data should not be required by the end user. This can also be supported by the two-level abstraction mechanism [TRAI82a]. However, there is some disagreement over whether the application programmer or the operating system should control the placement of data at a particular site. Also, if the location of data is known, then it should be possible to use this information to improve performance.

3.7 *Nested Transactions*

The original transaction concept did not provide the ability to nest subtransactions within a transaction in a hierarchical manner. However, the nested transaction model provides a more flexible abstraction for many applications [GRAY79]. Since subtransactions may fail independently of the parent, the parent may survive by retrying another subtransaction. Moss [MOSS82], Reed [REED78] and others have proposed ways to extend concurrency control schemes to nested transactions systems, but these have not yet been implemented and evaluated in a general purpose system.

3.8 *Distributed Directory/Dictionary Management*

General purpose distributed database management systems require a directory/dictionary (catalog) to help manage the database [LIND80a]. The directory/dictionary contains the definitions of the physical and logical structure of the data as well as the rules for mapping between the two. The directory also contains the local names of all resources - relations, files, programs, nodes, etc. - and the addressing rules for locating them. It can be thought of as a specialized distributed database system and as such can be implemented in a wide variety of ways: redundant vs. nonredundant copies, centralized vs. partitioned, etc.

The directory/dictionary problem is not unique to DDBMS's. The problems of defining, naming, and locating objects is central to the distributed programming environment [OPPE81] and has been discussed in Section 2.

3.9 *Typical Applications*

Real-time, interactive transaction processing is the "typical" application for a general purpose DDBMS. But the characteristics and requirements of these systems are vague. How much data is required in the entire database? How much data is required by a single transaction? What percent of the data is located locally vs. remotely? How many sites will a transaction need to access? What percent of transactions are read-only vs. update? What is the probability of conflict with other concurrent transactions? The lack of adequate answers to these and many other question makes it impossible to define a "typical" workload and application.

3.10 *Prototypes and Testbeds*

The best known DDBMS prototypes are SDD-1 [ROTH80], distributed INGRES [STNE77], and R* [LIND80b]. However, none of these systems meets all of the goals of a DDBMS [TRAI82a] and little has been published regarding the operational performance of these systems.

In order to compare and understand the tradeoffs involved in the design DDBMS's, a more flexible experimental approach is needed. These systems, called *testbeds*, emphasize modularity and flexibility so different algorithms and strategies can be easily "plugged in" and compared. They must also be architecturally similar to real systems in order for the experimental results to be meaningful. The DDTS project at Honeywell [ELHA81] is an

example of a planned testbed system for DDBMS experimentation on a wide variety of issues: data models, multischema architectures, user interfaces, semantic integrity, data translation, data allocation, transaction optimization, concurrency control, and reliability and recovery. The CARAT project at DEC/UMASS [GARC83] is an operational testbed system where the focus is on the issues of distributed concurrency control, deadlock detection, and crash recovery.

4.0 *CRITICAL RESEARCH ISSUES*

Identifying critical research issues is a difficult task because presenting them at too high or low a level renders them meaningless. Therefore, we attempt to present the issues at an intermediate level accompanied by some specific examples of open questions.

Broadly speaking, further research needs to be carried out in all the areas we have dealt with in detail earlier. The current state of the art is such that different proposals exist for solving, albeit partially, some of the problems, but not much is known about their appropriateness, efficiency and applicability in a distributed environment and the tradeoffs that they entail. Hence, there is a need for the following:

1 *Theory:* A theory of distributed systems needs to be developed in order to deal with issues such as complexity, theoretical limitations, and semantics. Formalisms for distributed computation (both for specifications and for analysis) are required. These formalisms should be able to handle failure-prone systems also, since error-recovery is one of the crucial aspects of distributed systems.

2 *Specification:* There is a need to design languages that provide for the specification of a number of features which were, so far, entirely the responsibility of the underlying operating system. These features include data and control distribution, choice of communication primitives, protection requirements and error-recovery.

3 *Design, Experimentation and Evaluation:* Design methodologies are needed for distributed systems. There is also a need to emphasize the *integration* of the various solutions to distributed system problems. The experiments should be geared towards building a core of knowledge pertaining to the issues that are relevant to such systems. One of the motivations behind these experiments should be to obtain performance measurements that can be used to evaluate the different proposals. Thus modelling, simulation, and proof techniques should be developed to analyze and evaluate the experimental systems. Obviously, not every solution will be appropriate for all situations. Hence the evaluation of the various proposals should also be directed towards uncovering the assumptions under which a particular scheme performs well. This is especially important in practice, since in the future, distributed systems are bound to be used both for special-purpose as well as general applications.

We now list some of the important open questions that should be addressed in terms of theory, specification, design, experimentation and evaluation.

1 *Distribution of Control:* Decentralized control algorithms for various functions of operating systems are needed, especially those concerned with a high degree of cooperation between decentralized controllers. Investigation of scheduling concepts such as bidding, clustering, coscheduling, pause time, wave scheduling is required. The use of various mathematical models such as adaptive control, stochastic control, statistical decision theory, and stochastic learning automata for dealing with uncertainty, inaccuracies and delay in distributed systems is also necessary.

2 *Distribution of Processes and Resources:* It is an open question on how to distribute processes that cooperate to execute a given task. This would affect the topology of the resulting network of processes and the manner in which individual nodes are designed. A crucial question is whether movement of processes in execution is worth it and what the best means to implement such movement is. Tradeoffs between static and dynamic allocation of resources should be investigated. Directory assignment, replication and partitioning should also be addressed. For client-server models of distributed file systems, where do we divide the responsibility between file servers and clients? When should distributed file systems be embedded in the operating system and when should a file server model be used? How should the operating system itself be distributed?

3 *Protection and Security:* Specification techniques for the protection requirements within a node and between nodes are needed. Models for protection are required. The problems with revocation of access rights in a distributed environment must be solved. Schemes for implementing protection requirements in a communicating failure-prone environment are needed. Encryption of messages for secure communication and integration of security and protection must be further addressed.

4 *Inter-Process Communication:* Specification of various aspects of communication paths, such as, directionality, structure of messages, ownership and access rights should be possible. Given the usefulness of different types of message primitives, users should be able to choose and specify the one appropriate for a given situation. Since there are a number of schemes for addressing processes and resources, each requiring differing operating system support and each providing differing levels of transparency and flexibility, the efficacy of the addressing schemes needs further investigation.

5 *Error Recovery:* A more comprehensive theory as well as realistic and practical schemes for error recovery are required. Overheads introduced by fail-safe systems and the trade-offs that such schemes entail must be investigated. How effective is decentralized control as an aid in providing error recovery? Is the nested atomic action the appropriate programming abstraction for the high level programmer? Can it be efficiently supported? Issues related to network partitioning and slow degradation of the working of nodes in a network is a fairly

recent but important research topic.

6 *Heterogeneity:* Techniques for dealing with information transfer between nodes with varying capabilities, storage schemes, and datamodels are needed. Efficient network interfaces for heterogenous hosts must be developed. How does heterogeneity affect the distribution of programs and data in a distributed system and vice versa?

7 *Synchronization and Concurrency:* In order to implement the atomic action abstraction, some system level mechanism for synchronizing concurrent access requests to shared data must be provided. Concurrency control theory is well developed, and there are many algorithms and approaches for solving the associated distributed deadlock problem assuming locking is used, but there is no sound basis for choosing one approach over another. Experimental studies will be required to compare the performance of alternate approaches for a variety of applications.

Bibliography

[ADA80] "Reference Manual for the Ada Programming Language," U.S. Department of Defense, July 1980.

[AGRA82] Agrawala, A.K., S.K. Tripathi, and G. Ricart, "Adaptive Routing Using a Virtual Waiting Time Technique," *IEEE Transactions on Software Engineering*, Vol. SE-8, No. 1, January 1982.

[ANDE71] Anderson, B., et al., "Data Reconfiguration Service," Technical Report, Bolt Beranek and Newman, May 1971.

[ANDR81] Andrews, G.R. "Synchronizing Resources," *ACM Transactions on Programming Languages and Systems* 3, 4, October 1981, 405-430.

[AOKI78] Aoki, Masanao, "Control of Large-Scale Dynamic Systems by Aggregation," *IEEE Transactions on Automatic Control*, June 1978.

[APPO81] "Apollo Domain Architecture," Apollo Computer, Inc., February 1981.

[APT80] Apt, K.R., N. Francez, and W.P De Roever, "A Proof System for Communicating Sequential Processes," *ACM Transactions on Programming Languages and Systems* 2, 3, July 1980, 359-385.

[BACH79] Bach, Maurice, Nancy Coguen, and Michael Kaplan, "The ADAPT System: A Generalized Approach Towards Data Conversion," *Proceedings of the Fifth International Conference on Very Large Data Bases*, Rio de Janeiro, Brazil, October 1979.

[BALL76] Ball, J.E., J. Feldman, J. Low, R. Rashid, and P. Rovner, "RIG, Rochester's Intelligent Gateway: System Overview," *IEEE Transactions on Software Engineering*, Vol. SE-2, No 4, December 1976.

[BALL76] Ball, J.E., J.A. Feldman, F.R. Cow, R.F. Rashid, and P.P. Rovner, "RIG, Rochester's Intelligent Gateway: System Overview," *IEEE Transactions on Software Engineering*, Vol. 2, No. 4, December 1976.

[BALT82] Balter, R., P. Berard, "Wh Control of Concurrency Level in a Distributed Systems is More Fundamental Than Deadlock Management," *ACM SIGACT-SIGOPS Symp. on Principles of Distributed Computing*, Ottawa, Canada, August 1982.

[BALZ71] Balzer, R.M., "Ports -- A Method for Dynamic Interprogram Communication and Job Control," Report for an ARPA contract at RAND, August 1971.

[BARB81] Barbara, D. and H. Garcia-Molina, "How Expensive is Data Replication: An Example," Technical Report 286, Dept. of Electrical Engineering and Computer Science, Princeton University, June 1981.

[BASK80] Baskett, Forest, Andreas Bechtolsheim, Bill Nowichi, John Seamons, "The SUN Workstation: A Terminal System for the Stanford University Network," CS Dept., Stanford University, March 1980.

[BASK77] Baskett, F., J. Howard, J. Montague, "Task Communication in DEMOS," *Proceedings 6th ACM Symposium on Operating System Principles*, pp. 23-31, November 1977.

[BASK32] Baskett, F., J.H. Howard, and J.T. Montague, "TASK Communication in DEMOS," *Proc. 6th Symposium on Operating Systems Principles*, SIGOPS, 1977, pp. 23-32.

[BENN82] Bennett, John, "A Comparative Study of Four Object-Oriented Systems," Technical Report, Univ. of Washington, Seattle, 1982.

[BERN79] Bernstein, P.A., D.W. Shipman, and W.S. Wong, "Formal Aspects of Serializability in Database Concurrency Control," *IEEE Transactions on Software Engineering*, Vol. SE-5, No. 3, May 1979, pp. 203-216.

[BERN80a] Bernstein, P.A., and N. Goodman, "Concurrency Control in Distributed Database Systems," *ACM Computing Surveys*, Vol. 13, No. 2, June 1981, pp. 185-221.

[BERN80b] Bernstein, P.A., D.W. Shipman, and J.B. Rothnie, Jr., "Concurrency Control in a System for Distributed Databases (SDD-1)," *ACM Transactions on Database Systems*, Vol. 5, No. 1, March 1980, pp. 18-25

[BERN81a] Bernstein, P., and N. Goodman, "Concurrency Control in Distributed Database Systems," *ACM Computing Surveys*, Vol. 13, No. 2, June 1981.

[BERN81b] Bernstein, P., et al., "Query Processing in a Distributed Database (SDD-1)," *ACM Tran. on Database Systems*, Vol. 6, No. 4, December 1981.

[BERN82a] Bernstein, P., and N. Goodman, "A Sophisticate's Introduction to Distributed Database Concurrency Control," Research Report, TR-19-82, Harvard University, also

8th Intl. Conference on Very Large Data Bases, September 1982.

[BERN82b] Bernstein, P.A., and N. Goodman, "Concurrency Control Algorithms for Multiversion Database Systems," *ACM Symp. on Principles of Distributed Computing*, Ottawa, Canada, August 1982.

[BERS80] Berstis, V., "Security and Protection of Data in the IBM System/38," *Proceedings of the 7th Annual Symposium on Computer Architecture*, May 1980.

[BHAR82b] Bhargava, B., "Performance Evaluation of the Optimistic Approach to Distributed Database Systems and Its Comparison to Locking," *IEEE 3rd Intl. Conf. on Distributed Computer Systems*, October 1982.

[BIRR81] Birrell, Andrew, Roy Levin, Roger Needham and Michael Schroeder, "Grapevine: An Exercise in Distributed Computing," *Proc. of the Eighth Symposium on Operating System Principles*, December 1981.

[BOCH78] Bochman, G., "Finite State Description of Communication Protocols," *Computer Networks*, 2, 1978, 361-372.

[BOKH79] Bokhari, S. H., "Dual Processor Scheduling with Dynamic Reassignment," *IEEE Transactions on Software Engineering*, Vol. SE-5, No. 4, July 1979.

[BRIN75] Brinch Hansen, P., "The Programming Language Concurrent Pascal," *IEEE Transactions on Software Engineering*, Vol. SE-1 2, June 1975, 199-207.

[BRIT80] Britton, D.E., M.E. Stickel, "An Interprocess Communication Facility for Distributed Applications," *Proceedings 1980 COMPCON Conference on Distributed Computing*, February 1980.

[BRYA81] Bryant, R.M., R.A. Finkel, "A Stable Distributed Scheduling Algorithm," *Proc. 2nd International Conference in Distributed Computing Systems*, April 1981.

[CASE08] Casey, L. and N. Shelness "A Domain Structure for Distributed Computer System," *Proceedings of the 6th ACM Symposium on Operating Systems Principles*, November 1977, pp. 101-108.

[CHAN69] Chandrashekaran and Shen, "Stochastic Automata Games," *IEEE Transactions on Systems, Science and Cybernatics*, April 1969.

[CHAN79] Chandy, K.M. and J. Misra, "Deadlock Absence Proofs Networks of Communicating Processes," *Information Processing Letters*, November 1979, 185-189.

[CHAN81] Misra, J. and K.M. Chandy, "Proofs of Networks of Processes," *IEEE Transactions on Software Engineering*, SE-7, 4, July 1981, 417-426.

[CHAN82] Chandy, K.M., and J. Misra, "A Distributed Algorithm for Detecting Resource Deadlocks in Distributed Systems," *ACM Symp. on Principles of Distributed Computing*, Ottawa, Canada, August 1982.

[CHNG79] Chang, E., "An Introduction to Echo Algorithms," *Proc. First International Conference on Distributed Computing Systems*, 1979, 193-198.

[CHER79] Cheriton, D.R., M.A. Malcoim, L.S. Melen, G.R. Sager, "Thoth, a Portable Real-Time Operating System" *Communications of the ACM*, Vol. 22, No. 2, February 1979.

[CHON75] Chong, Chee-Yee, and Michael Athans, "On the Periodic Coordination of Linear Stochastic Systems," *Proceedings 1975 IFAC*, August 1975.

[CHOW79] Chow, Yuan-Chien, and Walter Kohler, "Models for Dynamic Load Balancing in a Heterogeneous Multiple Processor System," *IEEE Transactions on Computers*, Vol. C-28, No. 5, May 1979.

[CHOW82] Chow, T.C.K., and J.A. Abraham, "Load Balancing in Distributed Systems," *IEEE Transactions on Software Engineering*, Vol. SE-8, No. 4, July 1982.

[CHU69] Chu, W.W., "Optimal File Allocation in a Multiple Computing System," *IEEE Transactions on Computers*, Vol. C-18, pp. 885-889, October 1969.

[CHU80] Chu, W.W., L.J. Holloway, M. Lan and K. Efe, "Task Allocation in Distributed Data Processing," *IEEE Computer*, Vol. 13, pp. 57-69, November 1980.

[CHU82] Chu, W.W., and P. Hurley, "Optimal Query Processing for Distributed Database Systems," *IEEE Trans. on Computers*, Vol. C-31, No. 9, September 1982, pp. 835-850.

[COMP82] Special Issue on Data Flow Systems, *IEEE Computer*, February 1982.

[COOK79] Cook, R.P., "*Mod--A Language for Distributed Programming," *Proc. First International Conference on Distributed Computing Systems*, October 1979, 233-241.

[COSS74] Cosserat, D.C., "A Capability Oriented Multi-Processor System for Real-Time Applications," *Proceedings of the International Conference on Computer Communications*, October 1972.

[COX81] Cox, G., W. Corwin, K. Lai, F. Pollack, "A Unified Model and Implementation for Interprocess Communication in a Multi- Processor Environment," Intel Corporation, 1981.

[DAVI81] Davies, D.W., E. Holler, E.D. Jensen, S.R. Kimbleton, B.W. Lampson, G. LeLann, K.J. Thurber and R.W. Watson, *Distributed Systems--Architecture and Implementation*, Lecture Notes in Computer Science, Vol. 105, Springer-Verlag, NY, 1981.

[DENN66] Dennis, J., and E. Van Horn, "Programming Semantics for Multiprogrammed Computations," *Communications of the ACM*, Vol. 9, No. 3, March 1966.

[DENN76] Denning, P.J., "Fault-Tolerant Operating Systems," *Computing Surveys*, Vol. 8, No. 4,

December 1976, pp. 359-390

[DIO80] Dio, Jeremy, "The Cambridge File Server," *ACM, Operating System Review*, October 1980.

[ECKH78] Eckhouse, R.H., Jr., and J.A. Stankovic, "Issues in Distributed Processing - An Overview of Two Workshops," *IEEE Computer*, Vol. 11, No. 1, January 1978, pp. 22-26.

[EFE82] Efe, Kemal "Heuristic Models of Task Assignment Scheduling in Distributed Systems," *IEEE Computer*, Vol. 15, No. 6, June 1982.

[ELDE80] El-Dessouki, O.I., and W.H. Huan, "Distributed Enumeration on Network Computers," *IELE Transactions on Computers*, Vol C-29, 1980, pp. 818-825.

[ELLI77] Ellis, C., "A Robust Algorithm for Updating Duplicate Databases," *Proc. Second Berkeley Workshop on Distributed Management of Data and Computer Networks*, Berkeley, California, May 1977, pp. 146-158.

[ELMA81] Elmasri, R., C. Devor, S. Rahimi, "Notes on DDTS: An Apparatus for Experimental Research in Distributed Database Management System," *ACM SIGMOD Record*, Vol. 11, No. 4, July 1981, pp. 32-49.

[ENSL78] Enslow, P., "What is a Distributed Data Processing System," *IEEE Computer*, Vol. 11, No. 1, January 1978.

[ENSL80] Enslow, Philip and Timothy Saponas, "Distributed and Decentralized Control in Fully Distributed Processing Systems," Final Technical Report, GIT-ICS-81/82, September 1980.

[ESWA76] Eswaran, K.P., J.N. Gray, R.A. Lorie and I.L. Traiger, "The Notions of Consistency and Predicate Locks in a Database System," *Communications of the ACM*, Vol. 19, No. 11, November 1976, pp. 624-633.

[FARB73] Farber, D.J., et al., "The Distributed Computer System," *Proceedings 7th Annual IEEE Computer Society International Conference*, February 1973.

[FELD79] Feldman, J.A., "High Level Programming for Distributed Computing," *Communications of the ACM* 22, 6, June 1979, 353-368.

[FORS78] Forsdick, H.C., R.E. Schwartz and R.H. Thomas, "Operating Systems for Computer Networks," *IEEE Computer*, Vol. 11 No. 1, January 1978.

[FRID81] Fridrich, M., and W. Older, "The FELIX File Server," *ACM SIGOPS, Proceedings of the Eighth Symposium on Operating System Principles*, pp. 37-44, December 1981.

[FU70] Fu, King-Sun, "Learning Control Systems," *IEEE Transactions on Automatic Control*, April 1970.

[GALL77] Gallager, R., "A Minimum Delay Routing Algorithm Using Distributed Computation," *IEEE Transactions on Communications*, Vol COM-25, No. 1, January 1977.

[GARC78] Garcia-Molina, H., "Performance Comparison of Two Update Algorithms for Distributed Databases," *Proc. Third Berkeley Workshop on Distributed Data Management and Computer Networks*, Berkeley, 1978, pp. 108-119.

[GARC79a] Garcia-Molina, H., "A Concurrency Control Mechanism for Distributed Databases Which Uses Centralized Locking Controllers," *Proc. 4th Berkeley Conference on Distributed Data Management and Computer Networks*, August 1979, pp. 113-124.

[GARC79b] Garcia-Molina, H., "Performance of Update Algorithms for Replicated Data in a Distributed Database," Ph.D. Dissertation, Stanford University, 1979.

[GARC82] Garcia-Molina, H., "Reliability Issues for Fully Replicated Distributed Databases," *IEEE Computer*, Vol. 16, No. 9, September 1982, pp. 34-42.

[GARC83] Garcia-Molina, H., F. Germano, Jr., and W.H. Kohler, "Architectural Overview of a Distributed Software Testbed," *Proc. Sixteenth Hawaii Intl. Conf. on System Science*, Jan 1983. (to appear)

[GELE82] Gelernter, D. and A.J. Bernstein, "Distributed Communication via Global Buffer," *Proc Symposium on Principles of Distributed Computing*, August 1982

[GENT82] Gentlemen, W. M., "Message Passing Between Sequential Processes: The Reply Primitive and the Administrator Concept," *Software - Practice and Experience*, Vol. 11, pp. 435-466, 1981.

[GIFF79a] Gifford, D., "Weighted Voting for Replicated Data," *7th Symp. on Operating System Principles*, December 1979, pp. 150-159.

[GIFF79b] Gifford, D.K., "Violet: An Experimental Decentralized System," *Operating Systems Review* 13, 5, December 1979.

[GLIG80] Gligor, V.D., and S.H. Shattuck, "On Deadlock Detection in Distributed Systems," *IEEE Transactions on Software Engineering*, Vol. SE-6, No. 5, September 1980, pp. 435-440.

[GLOR80] Glorioso, Robert M., and Fernando Colon Osorio, *Engineering Intelligent Systems*, Digital Press, Bedford, Maryland, 1980.

[GOLD80] Goldberg, Adele, et al., "Smalltalk-80: The Language and Its Implementation," to appear.

[GONZ77] Gonzalez, M.J., "Deterministic Processor Scheduling," *ACM Computing Surveys*, Vol. 9, No. 3, pp. 173-204, September 1977.

[GOOD79] Good, D.I., R.M. Cohen, and J. Keeton-Williams, "Principles of Proving Concurrent Programs in Gypsy," *Proc. Sixth ACM Symposium on Principles of Programming Languages*," January 1979, 42-52.

[GRAY75] Gray, J.N., R.A. Lorie, and G.R. Putzolu, "Granularity of Locks in a Shared Database," *Proc. Intl. Conference on Very Large Databases*, September 1975, pp. 428-451.

[GRAY79] Gray, J.N., "Notes on Data Base Operating Systems," in *Operating Systems: An Advanced Course*, R. Bayer, R.M. Graham and G. Seegmuller, editors, Springer-Verlag, 1979, pp. 393-481.

[GRAY81] Gray, J.N., "The Transaction Concept: Virtues and Limitations," *VLDB*, September 1981, pp. 144-154.

[GUIL82] Guillemont, Marc, "The Chorus Distributed Operating System: Design and Implementation," International Symposium on Local Computer Networks, Florence, Italy, April 1982.

[HALT71] Halter, A.N., and G.W. Dean, *Decisions Under Uncertainty*, South-Western Pub. Co., Chicago, Illinois, 1971.

[HAMI78] Hamilton, J., "Functional Specification for the WEB Kernel," Digital Equipment Corporation, R & D Group, Maynard, Massachusetts, November 1978.

[HAMM78] Hammer, M., and D. Shipman, "An Overview of Reliability Mechanisms for a Distributed Database System," *COMPCON Spring 1978 Proceedings*, February 1978, pp. 63-65.

[HAMM80] Hammer, M., and D. Shipman, "Reliability Mechanisms for SDD-1: A System for Distributed Databases," *ACM Tran. on Database Systems*, December 1980.

[HAMM81] Hammer, M., and D. McLeod, "Database Description with SDM: A Semantic Database Model," *ACM Trans. on Database Systems*, Vol. 6, No. 3, September 1981, pp. 351-386.

[HO80] Ho, Y., "Team Decision Theory and Information Structures," *Proceedings of the IEEE*, Vol. 68, No. 6, June 1980.

[HOAR78] Hoare, C.A.R., "Monitors: An Operating System Structuring Concept," *Communications of the ACM*, 17, 10, October 1974, 549-557.

[IRAN79] Irani, K.B., and H. Lin, "Queuing Network Models for Concurrent Transaction Processing in a Database System," *SIGMOD* 1979.

[ISLO80] Isloor, Sreekaanth, and T. Angthony Marsland, "The Deadlock Problem: An Overview," *IEEE Computer*, Vol. 13, No. 9, 1980.

[JARV75] Jarvis, R.A., "Optimization Strategies in Adaptive Control: A Selective Survey," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SME-5, No. 1, January 1975.

[JENS78] Jensen, E.D., "The Honeywell Experimental Distributed Processor - An Overview of its Objective, Philosophy and Architectural Facilities," *IEEE Computer*, Vol. 11, No. 1, January 1978.

[JONE79] Jones, A.K., R.J. Chansler, I. Durham, K. Schwans, S.R. Vegdahl, "StarOS, a Multiprocessor Operating System for the Support of Task Forces," *Proceedings 7th Symposium on Operating System Principles*, December 1979.

[KAHN81] Kahn, K.C., et al., "iMax: A Multiprocessor Operating System for an Object-Based Computer," *Proceedings of the Eighth Symposium on Operating System Principles*, December 14-16, 1981.

[KAMO82] Kamoun, F., M.B. Djerad and G. Le Lann, "Queueing Analysis of the Ordering Issue in a Distributed Database Concurrency Control Mechanism: A General Case," *IEEE 3rd Intl. Conf. on Distributed Computed Systems*, October 1982.

[KATZ77] Katzman, J.A., "A Fault-Tolerant Computing System," Tandem Computers, Inc., Cupertino, California, 1977.

[KIMB78] Kimbleton, S.R., H.M. Wood and M.L. Fitzgerald, "Network Operating Systems--An Implementation Approach," *Proceedings AFIPS Conference*, 47, 1978.

[KLEI81] Kleinrock, L., and A. Nilsson, "On Optimal Scheduling Algorithms for Time-Shared Systems," *JACM*, Vol. 28, No. 3, pp. 477-486, July 1981.

[KNOT74] Knott, "A Proposal for Certain Process Management and Intercommunication Primitives," *Operating Systems Review*, October and January, 1974-1975.

[KOHL81] Kohler, W.H., "A Survey of Techniques for Synchronization and Recovery in Decentralized Computer Systems," *ACM Computing Surveys*, Vol. 13, No. 2, June 1981, pp. 149-183.

[KORT82] Korth, H.T., "Edge Locks and Deadlock Avoidance in Distributed Systems," *ACM SIGACT-SIGOPS Symp. on Principles of Distributed Computing*, Ottawa, Canada, August 1982, pp. 173-182.

[KUNG81] Kung, H.T., and J.T. Robinson, "On Optimistic Methods for Concurrency Control," *ACM Tran. on Database Systems*, Vol. 6, No. 2, June 1981.

[LAMP79] Lampson, B.W., and H.E. Sturgis, "Crash Recovery in a Distributed Storage System," revised and expanded unpublished paper, Computer Science Laboratory, Xerox Palo Alto Research Center, Palo Alto, California 94304, 1979.

[LAMP76a] Lampson, B.W., and H.E. Sturgis, "Reflections on an Operating System Design," *Communications of the ACM*, Vol. 19, No. 5, May 1976.

[LAMP76b] Lampson, B.W., and H.E. Sturgis, "Crash Recovery in a Distributed Storage System," unpublished paper, Computer Science Laboratory, Xerox Palo Alto Research Center, Palo Alto, California 94304, 1976.

[LANT80] Lantz, K.A., "RIG, an Architecture for Distributed Systems," *Proceedings ACM Pacific 80*, November 1980.

[LARS79] Larsen, R.E., *Tutorial: Distributed Control*, IEEE Catalog No. EFO 153-7, IEEE Press, New York, 1979.

[LAZO81] Lazowska, E., H. Levy, G. Almes, M. Fischer, R. Fowler, S. Vestal, "The Architecture of the Eden System," 8th Annual Symposium on OS Principles, December 1981.

[LELA78] Le Lann, G., "Algorithm for Distributed Data-Sharing Systems Which Use Tickets," *Proc. 3rd Berkeley Workshop on Dist. Databases and Comp. Networks*, 1978.

[LELA80] Le Lann, G., "Distributed Systems-Towards a Formal Approach," *Proceedings IFIP Congress*, Toronto, North Holland Pub., pp. 155-160, August 1980.

[LELA81] Le Lann, G., "A Distributed System for Real Time Transaction Processing," *IEEE Computer*, Vol. 14, No. 2, February 1981.

[LESS79] Lesser, V., D. Serrain, J. Bonar, "PCL: A Process-Oriented Job Control Language," *Proceedings 1st International Conference on Distributed Computing Systems*, October 1979.

[LESS80] Lesser, Victor, Daniel Serrain, Jeff Bonar, "PCL: A Process-Oriented Job Control Language," *IEEE Transactions on Computers*, Vol. C-29, No. 12, December 1980.

[LEVI77] Levine, Paul H., "Facilitating Interprocess Communication in a Heterogeneous Network Environment," Master's Thesis, MIT, June 1977.

[LEVY81] Levy, H., "A Comparative Study of Capability-Based Computer Architectures," University of Washington Master's Thesis, October 1981.

[LIN79] Lin, W.K., "Concurrency Control in a Multiple-Copy Distributed Database System," *4th Berkeley Conference on Distributed Data Management and Computer Network*, August 1979.

[LIN81] Lin, W.K., "Performance Evaluation of Two Concurrency Mechanisms in a Distributed Database System," *SIGMOD*, 1981.

[LIND79] Lindsay, B., et al., "Notes on Distributed Databases," *IBM Research Report*, Report RJ2571, 1979.

[LIND80a] Lindsay, B., "Object Naming and Catalog Management for a Distributed Database Manager," *IBM Research Report*, RJ2914 (36689), August 29, 1980.

[LIND80b] Lindsay, B., and P.G. Selinger, "Site Autonomy Issues in R*: A Distributed Database Management System," *IBM Research Report*, RJ2927 (36822), September 15, 1980.

[LIND81] Lindgren, B.W., *Elements of Decision Theory*, The MacMillan Company, New York, 1981.

[LISK75] Liskov, B., and S. Ziles, "Specification Techniques for Data Abstractions," *IEEE Trans. on Software Engineering*, Vol. 1, No. 1, March 1975, pp. 7-18.

[LISK79] Liskov, Barbara, "Primitives for Distributed Computing," *Proceedings 7th Symposium of Operating System Principles*, December 1979.

[LISK82] Liskov, B., and R. Scheifler, "Guardians and Actions: Linguistic Support for Robust, Distributed Programs," *Proc. Ninth Symp. on Principles for Programming Languages*, January 1982, 7-19.

[LISK82a] Herlihy, M., and B. Liskov, "A Value Transmission Method for Abstract Data Types," *ACM Transactions on Programming Languages and Systems*," Vol. 4, No. 4, October 1982, 527-551.

[LOME77] Lomet, D.B., "Process Structuring, Synchronization and Recovery Using Atomic Actions," *Proceedings ACM Conference on Language Design for Reliable Software, SIGPLAN Notices*, Vol. 12, No. 3, March 1977, pp. 128-137.

[LUDE81] Luderer, G.W.R., et al., "A Distributed UNIX System Based on a Virtual Circuit Switch," *Proceedings of the Eighth Symposium on Operating System Principles*, December 14-16, 1981.

[MA82] Ma, P.Y.R., E.Y.S. Lee and J. Tsuchiya, "A Task Allocation Model for Distributed Computing Systems," *IEEE Transactions on Computers*, Vol. C-31, No. 1, pp. 41-47, January 1982.

[MCQU74] McQuillan, J.M., "Adaptive Routing Algorithms for Distributed Computer Networks," BBN Report 2831, May 1974.

[MELL81] Schwartz, R.L. and P.M. Melliar-Smith, "Temporal Logic Specifications of Distributed Systems," *Proc. of the Second International Conference on Distributed Systems*, April 1981.

[MENA79] Menasce, D.A., and R.R. Muntz, "Locking and Deadlock Detection in Distributed Databases," *IEEE Trans. on Software Engineering*, Vol. SE-5, No. 3, May 1979.

[MENA80] Menasce, D.A., G.J. Popek, and R.R. Muntz, "A Locking Protocol for Resource Coordination in Distributed Databases," *ACM Trans. on Database Systems*, Vol. 5, No. 2, June 1980.

[MINO82] Minoura, T., and G. Wiederhold, "Resilient Extended True-Copy Token Scheme for a Distributed Database System," *IEEE Trans. on Software Engineering*, Vol. SE-8, No. 3, May 1982.

[MITC79] Mitchel, J.G., W. Maybury, and R. Sweet, "Mesa Language Manual," Xerox PARC Report CSL-79-3, April 1979.

[MOSS82] Moss, J.E.B., "Nested Transactions and Reliable Distributed Computing," *Second Symp. on Reliability in Distributed Software and Database Systems*, July 1982.

[NARE74] Narendra, K., "Learning Automata - A Survey," *IEEE Trans. on Systems, Man and Cybernetics*, Vol. SMC-4, No. 4, July 1974.

[NEED77] Needham, R.M., and R.D.H. Walker, "The Cambridge CAP Computer and its Protection System," *Proceedings 6th ACM Symposium on Operating System Principles*, November 1977.

[NELS81] Nelson, B.J., "Remote Procedure Call," Xerox Corporation Technical Report CSL-81-9, May 1981.

[OBER82] Obermarck, R., "Distributed Deadlock Detection Algorithm," *ACM Trans. on Database Systems*, Vol. 7, No. 2, June 1982.

[OPPE81] Oppen, D.C., and Y.K. Dalal, "The Clearinghouse: A Decentralized Agent of Locating Named Objects in a Distributed Environment," Xerox Corporation, Office Products Division Report OPD-T8103, October 1981.

[OUST80] Ousterhout, J., D. Scelza, P. Sindhu, "Medusa: An Experiment in Distributed Operating System Structure," *Communications of the ACM*, Vol. 23, No. 2, February 1980.

[OZSU82] Ozsu, M.T., and B.W. Weide, "Modeling of Distributed Database Concurrency Control Mechanisms Using a Generalized Petri-Net Formalism," *IEEE 3rd Intl. Conf. on Distributed Computed Systems*, October 1982.

[POPE81] Popek, G., et al., "LOCUS, A Network Trans. Parent, High Reliability Distributed System," *Proceedings of the Eighth Symposium on Operating System Principles*, December 14-16, 1981.

[POWE77] Powell, M., "The DEMOS File System," Proceedings 6th ACM Symposium on Operating System Principles, pp. 33-42, November 1977.

[RAIF61] Raiffa, H., and R. Schlaifer, *Applied Statistical Decision Theory*, Division of Research, Graduate School of Business Adm., Harvard University, Cambridge, Massachusetts, 1961.

[RAMA82] Ramamritham, K., "Proof Techniques for Resource Controllers," COINS Technical Report 82-18, University of Massachusetts, January 1982.

[RASH80] Rashid, R., "An Inter-Process Communication Facility for UNIX," Carnegie-Mellon University Technical Report, June 1980.

[RASH81] Rashid, R.F., and G.G. Robertson, "Accent: A Communication Oriented Network Operating System Kernel," *Proceedings of the Eighth Symposium on Operating System Principles*, December 14-16, 1981.

[RAO80] Rao, Ram, "Design and Evaluation of Distributed Communication Primitives," *ACM Pacific 1980*, November 1980.

[RASH81] Rashid, R., and G. Robertson, "Accent: A Communication Oriented Network Operating System Kernel," Carnegie-Mellon University Department of Computer Science Technical Report, April 1981.

[REED78] Reed, D.P., "Naming and Synchronization in a Decentralized Computer System," Ph.D. Dissertation, MIT, 1978.

[REED79] Reed, D.P., "Implementing Atomic Actions on Decentralized Data," *Proceedings 7th ACM Symposium on Operating System Principles*, December 1979.

[RIES79a] Ries, D., "The Effects on Concurrency Control on the Performance of a Distributed Data Management System," *Proc. 4th Berkeley Workshop on Distributed Data Management Comp. Network*, 1979.

[RIES79b] Ries, D.R., and M.R. Stonebraker, "Locking Granularity Revisited," *ACM Trans. on Database System*, June 1979, pp. 210-227.

[RITC74] Ritchie, D. and K. Thompson, "The UNIX Time-Sharing System," *Communications of the ACM*, Vol. 17, No. 7, July 1974.

[ROSE78] Rosenkrantz, D.J., R.E. Stearns and P.M. Lewis, "System Level Concurrency Control for Distributed Database Systems," *ACM Transactions on Database Systems*, Vol. 3, No. 2, June 1978, pp. 178-198.

[ROTH77] Rothnie, J.B., and N. Goodman, "A Survey of Research and Development in Distributed Database Management," *Proc. Third Intl. Conference on Very Large Databases*, 1977, pp. 48-62.

[ROTH80] Rothnie, J.B., Jr., P.A. Bernstein, S. Fox, N. Goodman, M. Hammer, T.A. Landers, C. Reeve, D.W. Shipman and E. Wong, "Introduction to a System for Distributed Databases (SDD-1)," *ACM Transactions on Database Systems*, Vol. 5, No. 1, March 1980, pp. 1-17.

[ROWE79] Rowe, L.A., and K.P. Birman, "Network Support for a Distributed Database System," *Proc. 4th Berkeley Conference on Distributed Data Management and Computer Networks*, 1979.

[ROWE82] Rowe, L.A., and K.P. Birman, "A Local Network Based on the UNIX Operating System," *IEEE Transactions on Software Engineering*, Vol. SE-8, No. 2, March 1982.

[RUSS80] Russel, D.L., "State Restoration in Systems of Communicating Processes," *IEEE Transactions on Software Engineering*, March 1980, 183-194.

[RYPK79] Rypka, D.J., and A.P. Lucido, "Deadlock Detection and Avoidance for Shared Logical Resources," *IEEE Transactions on Software Engineering*, Vol. SE-5, No. 5, September 1979, pp. 465-471.

[SALT78a] Saltzer, J.H., "Research Problems of Decentralized Systems with Largely Autonomous Nodes," *Operating System Review*, Vol. 12, No. 1, January 1978, pp. 43-52.

[SALT78b] Saltzer, J.H., "Naming and Binding of Outlets," *Operating Systems: An Advanced Course*, Springer-Verlag, 1978.

[SAND78] Sandell, N., R. Varaiya, M. Athans and M. Safonov, "Survey of Decentralized Control Methods for Large Scale Systems," *IEEE Transactions on Automatic Control*, Vol. AC-23, No. 2, April 1978.

[SCHO78] Schoch, J.F., "Inter-Network Naming, Addressing and Routing," *Compcon 78*, Spring 1978.

[SCHL81] Schlageter, G., "Optimistic Methods for Concurrency Control in Distributed Database Systems," *VLDB*, 1981.

[SEGA77] Segall, A., "The Modeling of Adaptive Routing in Data-Communication Networks," *IEEE*

Trans. on Communications, Vol. COM-25, No. 1, pp. 85-95, January 1977. (SKEE81) Skeen, D., and M. Stonebraker, "A Formal Model of Crash Recovery in a Distributed System," *Proc. Fifth Berkeley Workshop on Distributed Data Management and Computer Networks*, February 1981, pp. 129-142.

[SMIT80] Smith, G.R., "The Contract Net Protocol: High Level Communication and Control in a Distributed Problem Solver," *IEEE Transactions on Computers*, Vol. C-29, No. 12, December 1980.

[SOLO79] Soloman, M.H., and R.A. Finkel, "The Roscoe Distributed Operating System," *Proceedings 7th Symposium on Operating System Principles*, March 1979.

[SOLO78] Soloman, M.H. and R.A. Finkel, "Roscoe: A Multi-Microcomputer Operating System," *Proceedings of the Second Rocky Mountain Symposium on Microcomputers*, pp. 291-310, August 1978.

[STAN83] Stankovic, J.A., "A Heuristic for Cooperation Among Decentralized Controllers," to appear in *INFOCOM* 83, October 1983.

[STAN82] Stankovic, J.A., N. Chowdhury, R. Mirchandaney, I. Sidhu, "An Evaluation of the Applicability of Different Mathematical Approaches to the Analysis of Decentralized Control Algorithms," *Proceedings COMPSAC 82*, to appear November 1982.

[STAN82] Stankovic, John A., "Software Communication Mechanisms: Procedure Calls Versus Messages," *IEEE Computer*, Vol. 15, No. 4, April 1982.

[STAN81] Stankovic, John A., "ADCOS - An Adaptive, System Wide, Decentralized Controlled Operating System," University of Massachusetts Technical Report, November 1981.

[STAN79] Stankovic, J.A., and A. van Dam, "Research Directions in (Cooperative) Distributed Processing," *Research Directions in Software Technology*, MIT Press, Cambridge, Massachusetts, 1979.

[STEM82] Stemple, D., K. Ramaritham and S. Vinter, "Preliminary Design of a Port-Based Operating System," COINS Technical Report 82-24, Department of Computer and Information Sciences, University of Massachusetts, October 1982.

[STON77] Stone, H.S., "Multiprocessor Scheduling with the Aid of Network Flow Algorithms," *IEEE Trans. on Software Engineering*, Vol. SE-3, January 1977.

[STON78] Stone, H.S., "Critical Load Factors in Distributed Computer Systems," *IEEE Trans. on Software Engineering*, Vol SE 4, May 1978.

[STON78a] Stone, H.S., and S.H. Bokhari, "Control of Distributed Processes," *IEEE Computer*, Vol. 11, No. 7, pp. 97-106, July 1978.

[STNE77] Stonebraker, M., and E. Neuhold, "A Distributed Database Version of INGRES," *Proc. 1977 Berkeley Workshop on Distributed Data Management and Computer Networks*, May 1977 pp. 19-36.

[STNE79] Stonebraker, M., "Concurrency Control and Consistency of Multiple Copies of Data in Distributed INGRES," *IEEE Transactions on Software Engineering*, Vol. SE-5, No. 3, May 1979, pp. 180-194.

[STNE81] Stonebraker, M., "Operating System Support for Database Management," *Communications of the ACM*, Vol. 24, No. 7, July 1981, pp. 412-418.

[STUR80] Sturgles, H., J. Mitchell and J. Israel, "Issues in the Design and Use of a Distributed File System," *ACM Operating System Review*, pp. 55-69, July 1980.

[SUNS77] Sunshine, C., "Interprocess Communication Extensions for the UNIX Operating System: 1. Design Considerations," Rand Corporation Publication R-2064/1-AF, June 1977.

[SWAN77] Swan, R.J., S.H. Fuller and D.P. Siewiorek, "Cm: A Modular Multi-Microprocessor," *AFIPS Conference*, Vol. 46, NCC, 1977.

[TEN81b] Tenny, R.R., and N.R. Sandell, Jr., "Structures for Distributed Decision-Making," *IEEE Transactions on Systems, Man and Cybernetics*, Vol. SMC-11, No. 8, pp. 517-527, August 1981.

[TEN81b] Tenney, R.R., and N.R. Sandell, Jr., "Strategies for Distributed Decision-Making," *IEEE Transactions on Systems, Man and Cybernetics*, Vol. SMC-11, No. 8, pp. 527-538, August 1981.

[THOM79] Thomas, R.H., "A Majority Consensus Approach to Concurrency Control for Multiple Copy Databases," *ACM Transactions on Database Systems*, Vol. 4, No. 2, June 1979, pp. 180-209.

[TRAI82a] Traiger, I., et al., "Transactions and Consistency in Distributed Database Systems," *ACM Transactions on Database Systems*, Vol. 7, No. 3, September 1982.

[TRAI82b] Traiger, I., "Virtual Memory Management for Database Systems," *ACM Operating System Review*, Vol. 16, No. 4, October 1982, pp. 26-48.

[VERH78] Verhofstad, J.S.M., "Recovery Techniques for Database Systems," *Computing Surveys*, Vol. 10, No. 2, June 1978, pp. 167-195.

[WALD72] Walden, David C., "A System for Interprocess Communication in a Resource Sharing Computer Network," *Communications of the ACM*, Vol. 15, No. 4, April 1972.

[WARD80] Ward, S., "TRIX: A Network Oriented Operating System," *Proceedings COMPCON*, 1980.

[WATS79] Watson, R.W. and J.G. Fletcher, "An Architecture for Support of Network Operating System Services," *Proceedings 4th Berkeley Conference on Distributed Data Management Networks*, August 1979, pp. 18-50.

[WILK79] Wilkes, M.V., and R.M. Needham, *The Cambridge CAP Computer and its Operating System*, Elsevier North Holland, 1979.

[WINK72] Winkler, R.L., *Introduction to Bayesian Inference and Decision*, Holt, Rindhard & Winston, Inc., New York, 1972.

[WIRT77] Wirth, N., "Modula: A Language for Modular Programming," *Software - Practice and Experience*, Vol. 7, No. 1, January 1977, pp. 3-35.

[WITT80] Wittie, L., and Andre M. Van Tilborg, "MICROS, a Distributed Operating System for Micronet, a Reconfigurable Network Computer," *IEEE Transactions on Computers*, Vol. C-29, No. 12, December 1980.

[WULF74] Wulf, W., E. Cohen, W. Corwin, A. Jones, R. Levin, C. Pierson and R. Pollack, "HYDRA: The Kernel of a Multiprocessor Operating System," *Communications of the ACM*, Vol. 17, No. 6, 1974.

ISSUE SUMMARY FOR HIGHLY PARALLEL ARCHITECTURES

by

James C. Browne

University of Texas
Austin, Texas

1.0 *OVERVIEW*

This paper is a draft working paper for the Panel on Highly Parallel Architectures of the NSF Information Technology Workshop. The material given is largely taken from an earlier paper discussing the issues of parallel computing raised at the April 1982 Workshop on Parallel Computing held at NYU under the sponsorship of the DOE Laboratories.

The positions and opinions expressed herein are at this point entirely the responsibility of the author. There are several main themes. The first is that parallel computing is the issue and not just parallel architectures.

The potentially great advance in computational power available from parallelism will only be realized when the parallel architectures are clothed in all of the software, methods and algorithms and interfaces which will make them usable. The spectrum of problems in parallel computing spreads all the way from component technology to programming environments. The second major issue is the need to close the gaps between theory, architectures and applications.

It is now possible to design and implement powerful chip-level functional units based on specific algorithms with fixed dimensionality. It is vital to be able to construct models and theories of computing which make these building blocks truly useful to a broad spectrum of problems. The third main theme is that the principal reason for pursuing parallel architectures is to attain levels of performance which will enable us to explore and exploit applications of computers not feasible without very much higher performance.

This focus on performance is what distinguishes parallel computing from distributed computing which faces an essentially similar set of logical problems to parallel architectures. This mapping is very much more complex than mapping to sequential architectures. It may be the most difficult problem of all.

2.0 *SPECIFIC PROBLEM AREA*

2.1 *Components*

The ability to design and fabricate chip level elements has vastly outstripped the technology for packaging and integration. The crucial problems are distance, heat dissipation and communication. The communication problems are discussed in a general context in Section 3.4.

2.2 *Architectures*

There is a vast proliferation of proposals for highly parallel architectures. The crucial point is that no single classical model of parallel computation will be most effective for all problems. The mapping of problems to architectures simply cannot be accomplished in a reasonable manner for arbitrary problem-architecture pairs when the architecture has limited communication geometry. Thus, there will be requirements for architectures realizing a variety of models of parallel computation. The research problems include the determination of the applicability of problems to models of computation.

All parallel architectures will require trade-offs between synchronization costs, data communication costs and flexibility of the mappings supported. This is a research issue where experimentation is needed.

The memory and I/O systems for parallel architectures have received very little attention beyond simple theoretical models. Serious analyses and/or experimental work in these areas is almost totally lacking and is a probable short range bottleneck for highly parallel architectures. The necessity for many independent I/O streams and management of them have hardly been considered.

2.3 *Software*

Operating systems, programming languages and programming environments for many-processor architectures are in a very primitive state. There are a few multi-processor systems but little serious attention has been given to the problems of extrapolating their operating system architectures to large numbers of processors. The languages available for parallel programming have not had sufficient use to establish a basis for evaluation. Most of the languages with parallel programming constructs were designed for system programming and management of only a small number of processes.

Programming environments for parallel computing are almost only gleams in researchers' eyes. One critical question which must be addressed is migration from serial to parallel architectures. It is not plausible to expect a restart from scratch to apply parallel architectures.

2.4 *Algorithms And Problem Analysis*

This is again a critical issue. The need is not only for methods but for education. Parallel thinking is far more crucial than parallel architectures for effective application of parallel computing. This requirement spreads across all applications from expert systems to weapons design. It is the area where the longest time and greatest total effort will be required.

3.0 *GENERAL ISSUES FROM A SYSTEMS PERSPECTIVE*

Parallel computing adds three significant issues to those faced in sequential computing: communication between the parallel processes, synchronization/sequencing (S/S) among the parallel processes and problem decomposition

where the number of processing elements does not equal the dimension of the natural structure of the problem or algorithm. The first two of these issues have been much studied in the context of operating systems and more recently in the context of data base systems. There has been recent attention from computer science theorists from the validation/verification viewpoint. There has, however, been relatively little attention paid to these problems from the perspective of practical high-performance computing. The difference in perspective of research in parallel computing and in the previous work in other areas is rather amazing. There is substantial overlap between the three issues. Synchronization of sequencing is often obtained by binding data to processes and controlling sequences by requests for access to data. Data movement costs may be dominated by multiple transfers of data caused by awkward mappings of processing to processing elements. Each model of parallel computation is characterized by some choices made on these two issues. There are a number of other significant factors such as the size of the parallel execution unit, but these may be considered design parameters for an architecture rather than issues basic to the structuring of parallel computation.

One important characterization of parallel computing is the time that communication paths and synchronization mechanisms are bound to hardware. Another is the selection of choices available in a given architecture. The time of binding has a significant impact on the execution cost of both data movement and S/S. An early (hardware) binding of S/S usually implies low overhead. Run-time (software) S/S usually implies a higher overhead per synchronization action. An SIMD architecture is synchronized by explicit hardware clocking. The S/S overhead is built in to the basic cycle times of instruction execution and is not visible as "overhead". An MIMD architecture must, however, be synchronized at execution time since the times of interactions cannot be predicted. Dataflow models of computing, whether data driven or demand driven, map the synchronization and sequencing entirely to the communication problem.

Analysis of communication and data movement overhead is far more complex. There are two variables here; geometry of paths and circuit-type paths versus packet-type paths. Both geometry and routing must be bound to hardware at some point between hardware design and program execution. The endpoints of communication architecture are a fixed single geometry network and a shared memory architecture. A fixed geometry of paths between processing elements will give efficient unit transfers if the paths are circuits. The amount of data which must be moved to execute a given algorithm may, however, vary dramatically between path geometries. A poor match between architecture geometry and algorithm can lead to order of magnitude degradations in performance. There have recently begun some serious studies of this problem [GEN78, GR080, KAP81]. Packet switching offers greater flexibility than static circuit switching, is still restricted by network path structures, and pays an overhead cost in that a routing decision must be taken at each step through the network. A multiprocessor in which all processors share a common memory can implement any communication geometry but must pay its overhead in either resolving or avoiding memory conflict.

The problems of mapping processing to processors is common to all architectures with a number of processors between 1 and n when n is the largest number of computations which can be executed in parallel in an algorithm. It is illustrated by the data movement overhead incurred by a systolic array processor of dimension n for multiplying matrices of dimension m, $m >> n$.

We try to summarize the trade-offs made by some of the projects in the area of communications versus parallelism and synchronization/sequency versus parallelism.

There are two classes of problems; those which can be described as internal to the computational engine and those external to the computational engine. The internal problems are those discussed preceding: of synchronization, data movement, and mapping.

The external problems are those of data organization and data movement. The problems of feeding these almost incredibly powerful engines, or programming and of organizing them have been little faced. The projects that have made the most significant success have been ones in which most aspects of the total organization are inherent in the basic computation model of the architecture. This includes the data flow model and the systolic model.

A summary of the status of high-performance parallel computing might be that it is clearly possible to design and actually implement computational engines capable of executing a very large number of cycles, at least a factor of ten or so, or maybe a hundred above today's technology. The problem is that it is not possible to currently implement systems which will in fact deliver such cycles usefully to users.

A further qualification is that it is probably possible to build effective high-performance parallel engines only for tasks of limited dimensionality. It does seem clear that there can be built specialized SIMD architectures (such as systolic arrays) which will be very effective on problems of limited dimensionality. It seems to this writer that data flow and MIMD architectures hold more promise as generally applicable parallel architectures than do the specialized SIMD architecture but their promise also seems a bit further from realization because of the greater complexity in obtaining efficient solutions to communication and synchronization problems in these late binding architectures.

There is a need for comprehensive projects which take an integrated approach to the entire problem. These will require large scale funding, organizational skills, technical strength in software, architecture as well as hardware and engineering.

One problem area which needs immediate emphasis is an analytical approach to determination of the parallel structuring of significant algorithms for both mathematical and data-oriented modeling. There are two approaches. One is to map algorithms to specific architectures. This approach is the staple of parallel analysis. The other approach is to look for the "natural" parallel structure of significant algorithms. The third task is to search for algorithms with a high degree of "natural" parallelism. The knowledge derived from this work can have significant impact on the design of special-purpose architectures.

We would emphasize that this work needs to go beyond consideration of algorithms in isolation to consideration of significant problems. The MIT studies of SIMPLE and the weather codes and the Texas study of PIC code are models of this type of work.

It is the opinion of this writer that many parallel architecture projects would benefit from an infusion of software and application expertise and emphasis.

APPENDIX D
POSITION PAPER ON SOFTWARE PRODUCTIVITY

by

Alfred M. Pietrasanta

IBM Systems Research Institute
New York, New York

Introduction

The state-of-the-art of software productivity is primitive. Professionals intuitively believe that there is a wide range of individual and group productivity; that there are key variables which influence productivity; that productivity has not significantly changed in the last decade; that there is a critical need to improve productivity; that there is potential to achieve this improvement.

However, these beliefs are subjective and difficult to substantiate. There is virtually no objective evidence backed up by comprehensive data collection, controlled experimentation, comparative evaluation. The critically important field of software productivity needs to be pulled out of the dark ages of visceral opinion into the modern world of objective, scientific evaluation.

Productivity Variables

What makes software productivity so difficult to analyze is the extreme number of variables which influence productivity. There are product variables (e.g. size, complexity); product environment variables (e.g. requirements stability, interface controls, testing complexity); development process variables (e.g. planning rigor, support tools, computer support); people variables (e.g. individual experience and capability, management capability), to name a very few.

Productivity Measurement

Compounding this awesome multi-variable problem is the disconcerting fact that there is no standard definition of software productivity. Everyone defines it differently, and measures it differently. As a result, the few measurements which do exist cannot be compared. Even such "simple" parameters as lines of code, people months or dollars can have extremely wide variations based on what the measurer includes or excludes.

Productivity Improvement

From a strategic point of view, the direction is clear. Software development is still a heavily labor-intensive process. The productivity of labor-intensive processes can be improved by automation, by replacing human activities with tools and computers, by minimizing human errors which result in expensive defect detection and correction. Since the human, creative component of software development will never be eliminated, software productivity can also be improved by providing programmers with a support environment which will allow them to work at their optimal productivity.

The strategy is clear, but implementing the strategy is very hard. Every potential productivity improvement requires an investment, and management would like some quantifiable indication of return on investment. In the absence of data, investment becomes an act of faith rather than a business decision. Software professionals must step up to their responsibility to put software productivity on an objective, rational, quantifiable base. Given that base, we can move forward with sound investments in process tools and support to exploit the great potential which exists for software productivity improvement.

Areas of Investigation

Listed below are a few possibilities for investigation. Each of these attack some facet of the complex question of software productivity, attempting to highlight both the major inhibitors to improved productivity, and the areas of greatest potential for improvement.

1) *Individual Productivity*

Almost two decades ago, SDC ran a controlled experiment on twelve programmers, showing a 5 to 1 variation in program size, a 13 to 1 variation in program speed, and a 25 to 1 variation in coding hours (productivity). This is an immensely fruitful, but largely unexplored, research area. Some interesting questions: What is individual productivity variability; What causes the variation; What can be done about it?

2) *Product Variability*

A deceptively simple question, "how does productivity vary with program size" remains unanswered. Some data shows productivity to be constant with size; other data shows productivity decreasing with increasing size. Internal IBM data shows productivity increasing with increasing size.

Professionals intuitively know that there is a spectrum of program complexity, from "simple" programs to "complex" programs. They also intuitively believe that complexity is correlated with productivity. Internal IBM data shows a 3 to 1 variation in productivity, based on complexity. Given the fact that there is no standard definition for either productivity or complexity, this area could benefit from some research.

3) *Support Variability*

A. J. Thadani's article in the *IBM Systems Journal* (Vol. 20, No. 4, 1981) appears to offer great potential for productivity improvement. He shows that providing an interactive user with sub-second response time can improve the user's productivity (interactions per hour) by two to three times. This could indicate that one of the biggest inhibitors to improved productivity could be the amount of computer support backing up the interactive programmer.

4) *Process Automation*

The potential for productivity improvement through automated tools - particularly at the front end of the

development process - is significant, but largely subjective. Formal requirements definition, formal specifications, abstract prototypes, reusable design libraries, are a few of the technologies explored in modern software engineering. However, basic questions of cost and productivity tradeoffs with these technologies are not well understood. On the negative side there is the investment in technology development and people education; on the positive side is the potential decrease in testing and maintenance costs. These costs/productivity tradeoffs need quantification.

5) *Productivity and Quality*

An absolutely fundamental question is how productivity relates to quality. Some very preliminary IBM data shows that post-ship quality directly correlates with pre-ship productivity. Since quality and productivity are perhaps the two basic parameters of most interest to the programming profession, we need to better understand their interrelationship.