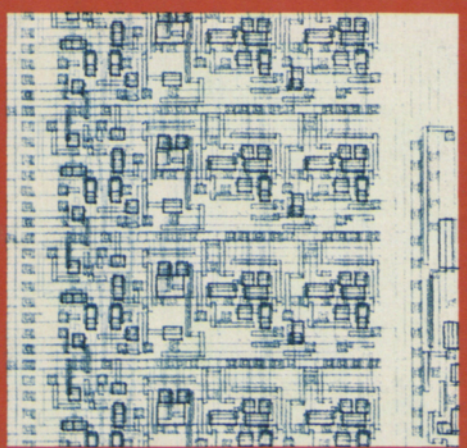
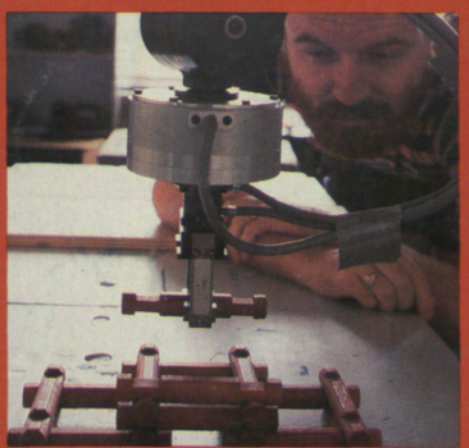
ENGINEERING

CORNELL QUARTERLY





VOLUME 20
NUMBER 2
AUTUMN 1985

TWENTY YEARS
OF
COMPUTER SCIENCE

IN THIS ISSUE

Engineering: Cornell Quarterly (ISSN 0013–7871), Vol. 20, No. 2, Autumn 1985.
Published four times a year, in summer, autumn, winter, and spring, by the College of Engineering, Cornell University, Campus Road, Ithaca, New York 14853. Second-class postage paid at Ithaca, New York, and additional offices. Subscription rate: $6.00 per year; $9.00 outside the United States.

Outside cover illustrations from Cornell research in computer science (clockwise from upper left): a small section of a computer image representing a VLSI design, part of a project in computer architecture; solid-model imaging—a gate valve assembly with blending surfaces shown in blue; a demonstration of computer-controlled robot assembly; an outer corridor of Cornell's projected performing arts center, as modeled by a researcher in the Program of Computer Graphics; a simple example of solid modeling with a blending surface.

Opposite: The terminal room in Upson Hall for graduate research. An addition to the building will provide less crowded facilities.

Gries

Reps and Teitelbaum

Schneider

Krafft

Solworth

Duke and Fish

TWENTY YEARS OF COMPUTER SCIENCE AT CORNELL

by David Gries

Twenty years ago Cornell became one of the first universities in the nation to create a computer science department. The initiative was unusual in several ways. It was helped by a grant from the Sloan Foundation. It was established initially as a graduate research program in order to begin educating the Ph.D.s who would populate the expected new departments in the new discipline. And because of the interdisciplinary nature of the field, the department was placed where it still is today, in both the College of Engineering and the College of Arts and Sciences.

In his first annual report the first chairman, Juris Hartmanis, wrote:

The major goal of this department is to become a Distinguished Department of Computer Science. We sincerely believe that computer science is a major new science, with an extensively broad range of influence, and that Cornell must excel in this area.

Twenty years later, we believe we are meeting that goal.

The most common way to measure such things is by peer rankings: in the 1982 survey conducted by the Conference Board of Associated Research Councils, Computer Science at Cornell placed fifth in faculty quality out of all the Ph.D.-granting departments—many considerably larger than ours—in the United States.

Another criterion is the effectiveness of the educational program: over the years, our department has produced 110 Ph.D.s, most of whom hold positions in academic institutions or industrial research laboratories. In research the department's program has expanded to include many areas of theoretical and experimental computer science, and research expenditures have increased more than three-fold in the last five years (see Figure 1).

These numbers do not, of course, tell the whole story. Priorities have been established and challenges met. Building an outstanding faculty and research program, developing courses and educational programs, acquiring facilities and keeping them up-to-date, and managing, somehow, to make room for an expanding operation are continuing goals.

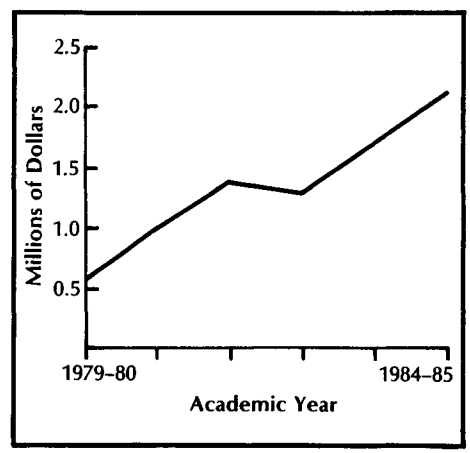


Figure 1. Annual expenditures for research in computer science at Cornell over the last six years. The graph shows the rise from $0.6 million to $2.1 million. The figures for 1984-85 do not include the donation by the Xerox Corporation of thirty workstations, related equipment, and maintenance service.

STARTING FROM SCRATCH IN BUILDING A FACULTY

The main faculty instigators at Cornell for establishing the Department of Computer Science were Dick Conway of the operations research faculty and

Table I.
FACULTY IN 1965

Richard W. Conway	Appointed jointly in operations research; now in the Johnson Graduate School of Management
Patrick C. Fischer	Now chairman of the computer science department at Vanderbilt University
Juris Hartmanis	
Christopher Pottle	Appointed jointly in electrical engineering; now fully in that school
Gerard Salton	
Sidney Saltzman	Appointed jointly in operations research; now in the Department of City and Regional Planning
Robert J. Walker	Appointed jointly in mathematics; now professor, emeritus

Bob Walker of the mathematics department; both initially had joint appointments in the new department. (Bob retired in 1974 from the mathematics faculty. Dick played a major role in the computer science department until just last year, when he moved to the Johnson Graduate School of Management at Cornell.)

In its first year, 1965–66, Computer Science had a faculty of seven (listed in Table I). Remarkably, five of the seven are still at Cornell, although only two are still with the department: Juris Hartmanis, the first chairman, and Gerard Salton, the second chairman. By the mid-1970s the faculty had doubled, and it remained at about fourteen members until the very late 1970s, when increasing interest in computer science, both locally and nationwide, forced the department to grow rapidly to its current size of twenty-four members (see Figure 2).

The recent growth has not been easy to accommodate. In a period of retrenchment within the University it has been difficult to get additional resources. The needs were documented, but the money was not always available. Also, the youth of the discipline—the oldest computer science department was established only in 1964—and its phenomenal growth have made it difficult to *find* faculty members, let alone to assimilate and retain them. Even now the demand for Ph.D.s is about four times the supply, which is about three hundred a year. The growth of the discipline is illustrated forcefully in the makeup of the Cornell department. The first retirement is perhaps ten years away, and the department consists of

In 1965 the initial computer science faculty at Cornell consisted of three full-time and four half-time people, as listed in Table I. Of the original seven, two are still in the department, five are still at Cornell, one is retired, and one is chairman of the computer science department at another university.

The scope of the department is suggested by Table II, which lists the current faculty members and the specialty areas in which they teach and conduct research. The youth of the field is illustrated by the fact that sixteen of the twenty-four professors—those whose names are printed in color—were appointed during the 1980s.

Table II.
FACULTY IN 1985

Özalp Babaoğlu: distributed systems, performance evaluation
Gianfranco Bilardi: VLSI algorithms and architectures, VLSI complexity, parallel computation
Ken Birman: distributed systems, fault tolerance, signal processing
Dina Bitton: databases
Tom Coleman: numerical analysis
Robert L. Constable: computational complexity, formal semantics, programming logics
John R. Gilbert: analysis of algorithms, combinatorial algorithms for numerical problems
David Gries: programming methodology, programming languages, compiler construction
Juris Hartmanis: theory of computation
John E. Hopcroft: algorithms, robotics
Greg Johnson: programming environments, compilers
Kevin Karplus: VLSI, computer music, computer-aided design
Dexter Kozen: computational complexity
Abha Moitra: programming methodology
Alex Nicolau: parallel computation, architecture, optimizing compilers
Prakash Panangaden: programming languages and logics, mathematical foundations of semantics
Gerard Salton: information organization and retrieval
Fred B. Schneider: concurrent programming, fault tolerance, distributed systems
Jon Solworth: VLSI design, computer-aided design, computer architecture
Ray Teitelbaum: programming languages and systems
Sam Toueg: computer networks and protocols, distributed computing
Charles Van Loan: numerical analysis
Vijay Vazirani: algorithms, computational complexity
Kay Wagner: applied logic

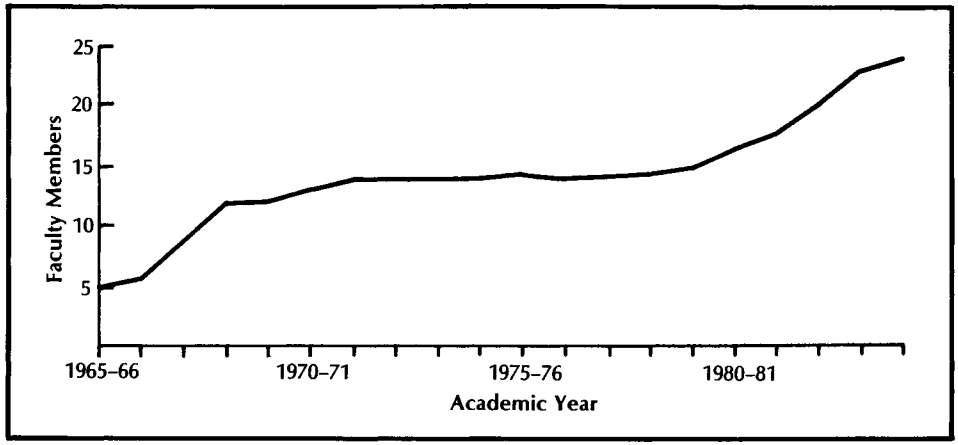


Figure 2. Growth in the computer science faculty at Cornell. Since its establishment in 1965, the faculty has increased almost five-fold, with much of the increase occurring during the past five years. The numbers refer to full-time-equivalent positions.

five full professors, four associate professors, fourteen assistant professors, and one lecturer.

A DISTINGUISHED FACULTY: THE FUNDAMENTAL ASSET

Regardless of its size, a department can only be as good as its individual faculty members, and in this regard Cornell's computer science department is clearly outstanding. Gerard Salton is an example. He received the first annual SIGIR (Special Interest Group in Information Retrieval) award in 1982 for long-lasting contributions to research in that area, and in 1983 the University of Helsinki recognized him for his research contributions. Another charter member of the department, Juris Hartmanis, was elected a member of the New York State Academy of Science in 1981, and that same year became the first computer scientist to receive an automatic extension to a National Science Foundation research grant under a program for special creativity. At Cornell Hartmanis was named the Walter R. Read Professor of Engineering in 1980. Another chairholder is John Hopcroft, named this year as the Joseph C. Ford Professor of Computer Science. David Gries was awarded a Guggenheim fellowship in 1983–84; a few years earlier he participated in the Distinguished Scholar Exchange Program, spending a month in China.

The younger faculty members have also received recognition. John Gilbert was a recipient this year of a National Science Foundation Presidential Young Investigator Award. Kevin Karplus, Fred Schneider, and Vijay Vazirani have all received IBM Faculty Development Awards under a program established just three years ago. And Özalp Babaoğlu was a recipient of the 1982 Sakrison Memorial Prize at Berkeley for his work in developing the Berkeley-UNIX operating system, which is the major system used in most computer science departments.

Faculty effectiveness is demonstrated also by the success of students. Cornell Ph.D.s have done well both in academia and in industrial research. In 1983, for example, Tom Reps received the second Doctoral Dissertation Award of the Association for Computing Machinery (ACM) for his thesis *Generating Language-Based Environments*, written under the direction of Tim Teitelbaum. In 1977 Susan Owicki and David Gries received the ACM Award for the Best Paper in Programming Languages and Systems; it was based on her Ph.D. thesis *Axiomatic Proof Techniques for Parallel Programs*.

Ever since the department's beginnings, the faculty has been prolific in publishing not only professional papers, but books and monographs (see Table III). In fact, more highly regarded texts appear to have originated at Cornell than at any other university. Many of these books have received wide acclaim and have been translated into other languages.

The department is well represented in professional activities at the national level. In a typical year it provides about twenty editors and members of editorial boards of respected journals and book series and eight to fifteen members of program committees for technical conferences. Working on editorial boards and program committees can be ardu-

Table III.
BOOKS BY MEMBERS OF THE COMPUTER SCIENCE FACULTY

1966 J. Hartmanis (with R. E. Stearns). *Algebraic structure of sequential machines.* Prentice-Hall.

1968 R. Conway, W. L. Maxwell (with L. W. Miller). *Theory of scheduling.* Addison-Wesley.

G. Salton. *Automatic information organization and retrieval.* McGraw-Hill.

P. Wegner. *Programming languages, information structures and machine organization.* McGraw-Hill.

1969 J. Hopcroft (with J. Ullman). *Formal languages and their relation to automata.* Addison-Wesley.

1971 D. Gries. *Compiler construction for digital computers.* John Wiley.

G. Salton. *The SMART retrieval system: Experiments in automatic document processing.* Prentice-Hall.

1973 R. Conway, D. Gries. *An introduction to programming: A structural approach.* Winthrop.

1974 J. Hopcroft (with A. V. Aho and J. D. Ullman). *The design and analysis of computer algorithms.* Addison-Wesley.

1975 G. Salton. *Dynamic information and library processing.* Prentice-Hall.

1976 R. Conway, D. Gries. *Primer on structured programming.* Winthrop.

R. Conway, D. Gries (with E. C. Zimmerman). *Primer on Pascal.* Winthrop.

J. E. Donahue. *Complementary definitions of programming language semantics.* Lecture notes on computer science, vol. 42. Springer-Verlag.

1977 R. Conway. *A primer on disciplined programming.* Winthrop.

R. Conway, D. Gries (with D. Wortman). *Introduction to structured programming using SP/k.* Winthrop.

1978 R. Constable. *A programming logic.* Winthrop.

R. Conway. *Programming for poets: A gentle introduction using PL/I.* Winthrop. [Other versions for FORTRAN, BASIC, and Pascal appeared in 1978, with J. Archer and R. Conway as coauthors.]

D. Gries, ed. *Programming methodology: A collection of articles by members of IFIP WG2.3.* Springer-Verlag.

1979 R. Cartwright. *A practical formal semantic definition and verification system for TYPED LISP.* Garland.

R. Conway, D. Gries (with C. Bass and M. Fay). *An introduction to microprocessor programming.* Winthrop.

1981 D. Gries. *The science of programming.* Springer-Verlag.

1982 R. Constable (with S. D. Johnson and C. D. Eichenlaub). *Introduction to the PL/CV2 programming logic.* Lecture notes in computer science, vol. 135. Springer-Verlag.

1983 J. Hopcroft (with A. V. Aho and J. D. Ullman). *Data structures and algorithms.* Addison-Wesley.

G. Salton (with M. J. McGill). *Introduction to modern information retrieval.* McGraw-Hill.

G. Salton, ed. (with H. J. Schneider). *Research and development in information retrieval.* Lecture notes in computer science, vol. 146. Springer-Verlag.

C. Van Loan (with G. Golub). *Advanced matrix computations.* The Johns Hopkins Press.

1984 T. Coleman. *Large sparse numerical optimization.* Lecture notes in computer science, vol. 165. Springer-Verlag.

F. B. Schneider (with six others). *Distributed systems: Methods and tools for specification.* Lecture notes in computer science, vol. 190. Springer-Verlag.

T. Reps. *Generating language-based environments.* MIT Press.

ous and time-consuming, but it is an important way to provide leadership and help ensure high quality in a new and growing field.

The faculty should grow to about thirty members in the next five or six years. Especially needed are senior professors, to bring more balance into the department. (This year some progress seemed to be made when Dexter Kozen, a theoretician, came from an industrial research laboratory to an associate professorship at Cornell; however, another associate professor, Alan Demers, was lost to industry.) The department also needs to increase the number of research associates (currently there are three). Such growth in faculty and research staff is essential not only because of the increasing teaching load, but also because of the need for more education and research in computer science at Cornell and throughout the United States. Research in computer science is an increasingly vital part of the nation's efforts to maintain technological and scientific leadership in today's "information society".

"Today more than half of all Cornell undergraduates... take at least one computer science course."

THE COMMITMENT TO UNDERGRADUATE EDUCATION

Although computer science at Cornell was originally conceived as a graduate program, the department has always taken its undergraduate teaching seriously. For example, in the early 1970s, drawing on strong faculty research interest in programming methodology, we assumed responsibility for teaching introductory programming to all engineering students. Today more than half of all Cornell undergraduates in the various colleges take at least one computer science course, and many nonmajors take junior- and senior-level courses simply because of their interest in the subject. The department performs a vital service for the whole University.

To help teach these courses, the department has continually developed appropriate software. The programming languages CORC and CUPL were introduced in the early 1960s. Dick Conway's PL/C compiler, the first error-correcting load-and-go compiler, was used heavily throughout the 1970s for teaching programming not only at Cornell, but in many other universities. Tim Teitelbaum's development of the Cornell Program Synthesizer on the Terak computer was the first serious effort to provide an interactive environment on microcomputers that was suitable for teaching large numbers of students. These contributions have placed Cornell at the forefront of the drive to provide urgently needed software aids for teaching programming.

The introductory programming courses are large. For example, the first course, CS100, has between 600 and 900 students each semester; the second course, CS211, has between 325 and 400. Generally, two faculty members are assigned to such a course, and they do the lecturing in large sections. Teaching assistants are in charge of recitations and optional lectures. Any student can get individual help from the teaching staff if that is necessary. And finally, undergraduate proctors—students who have been through the course—staff a consulting room for at least eight hours a day so that students can get help with their programs. We believe that these large courses are taught quite effectively—with our current resources there is no better way to teach them. Still, we feel

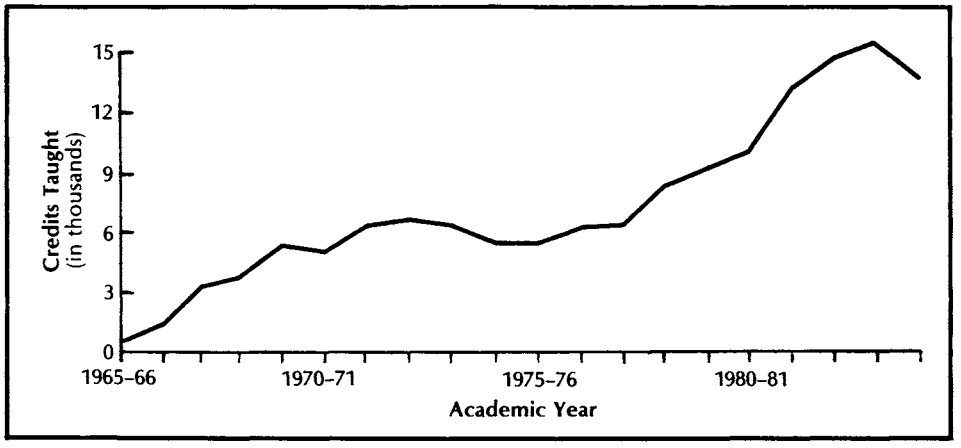


Figure 3. The explosive expansion in enrollments in computer science. The amount of instruction provided by the department is plotted in terms of the total number of credits earned by the students (a typical course gives three or four credits). In 1984-85 the number of credits averaged 578 per faculty member.

that the students would benefit from more individual attention than they are now getting. We would like to set up more recitation sections so that all the students could be accommodated in required sections of no more than thirty students each.

Our main problem is the tremendous growth in enrollments (see Figure 3). Since 1965-66, when instruction began, the number of courses taught by the department has increased eighteen-fold, and last year computer science professors had approximately twice the teaching load of the average College of Engineering faculty member.

Teaching in such a young field entails other problems. Each year new research uncovers new concepts, new ways of presenting material, and new software and hardware to teach with, and the continual change is felt at the undergraduate as well as the graduate level. Texts are often unsuitable and out of date. (In 1973 Conway and Gries wrote an introductory programming text that was used at Cornell for twelve years, but the switch to different machines and programming languages, as well as advances in the knowledge of programming, has made it obsolete.) And finally, the widespread use of computers in many academic areas besides computer science has created a whole new set of problems. These challenges tend to make teaching more exciting, of course, and give the faculty more opportunity to lead in the development of the field. The large number of textbooks written by Cornell computer science professors is evidence of this leadership.

Throughout the first ten years, the department felt it was best for a student interested in computer science to minor in that field and major in another area such as mathematics, physics, or linguistics. The thought was that in such a new field it would take time for the faculty to determine what concepts were important and lasting, and which should be taught in an undergraduate curriculum. Further, it was important to educate people in *other* fields to use computers intelligently. Accordingly, the department taught many undergraduate courses but gave no undergraduate degree.

Pressures for an academic major increased in the late 1970s, however, and in 1978 the department established *two* programs leading to undergraduate degrees: the Bachelor of Science in the College of Engineering and the Bachelor of Arts in the College of Arts and Sciences. The programs are going strong. In 1984-85 there were 220 computer science majors, and thirty-five B.S. and forty-six A.B. degrees were awarded. Last year the Association for Computer Science Undergraduates (ACSU) was formed on campus. More communication between students and the faculty has been promoted also through the efforts of our undergraduate administrator, Michele Fish. These initiatives have created a friendlier and more satisfactory environment for the undergraduates.

Unfortunately, the establishment and growth of the two undergraduate programs has created additional strain in the department, for the faculty has not grown in proportion to the workload. Our professors advise computer science majors from two colleges in programs that have different requirements. And

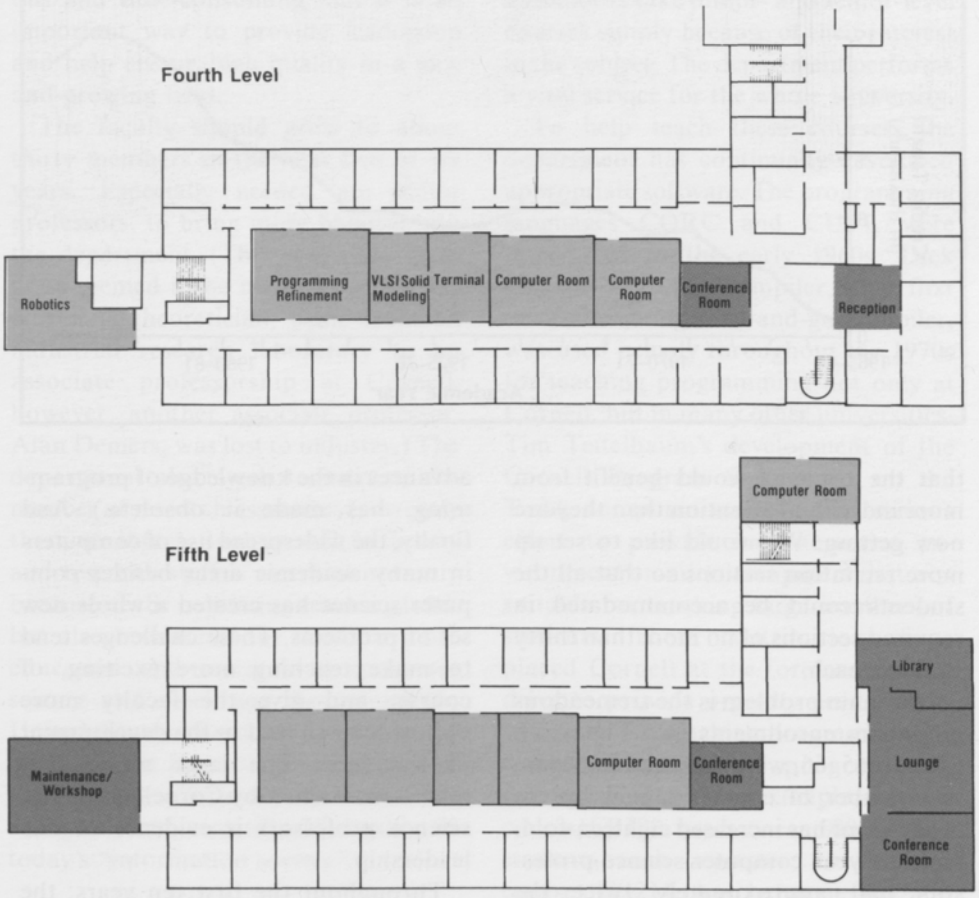


Figure 4. Floor plans for the addition to Upson Hall. Two floors will be built for the Department of Computer Science on the north-south wing, above the space now occupied by the School of Operations Research and Industrial Engineering. The lower of the two new floors will connect with the fourth floor of the east-west wing, currently occupied by the computer science department.

In the diagram, most of the rooms along the exterior walls are offices. Laboratories are shaded in gray, and special-purpose rooms in color.

of course the number of students in junior- and senior-level courses has increased and continues to rise; some 400-level courses have had more than one hundred students enrolled.

In his recent state-of-the-University address, Cornell President Frank H. T. Rhodes set forth as a major goal the provision of "diversified, distinctive, and distinguished undergraduate programs". In the efforts to achieve this goal, the computer and the science of computing should play important roles, the former as a tool and the latter as an aid in understanding how the tool might be used. An equally important function of computer science is to contribute exciting new scientific concepts through vigorous research activity. With adequate resources, the Cornell department intends to fulfill its responsibilities in reaching President Rhodes's goal.

MAKING ROOM FOR COMPUTER SCIENCE

In 1965 the Department of Computer Science was housed in a few offices on the north side of the fourth floor of Upson Hall. (The four large rooms on the south side were being used to teach drafting.) Since then, the department has expanded by encroaching on the space of others. Such expansion was not too difficult in the early 1970s, but it became a critical problem toward the end of the decade as the College buildings became more and more crowded. In his 1980–81 annual report, Chairman Hartmanis wrote that "the most urgent need for the department is *space*. Doubling our space would just barely take care of our needs for the next five years. Critical problems include the crowded conditions for the students, the lack of a conference room, and the lack of space for consulting for undergraduate courses." Although it was not recognized at the time, the accommodation of equipment as well as people mandated expansion. Over the past seven years, research in computer science at Cornell has changed from a pencil-and-paper operation to a laboratory enterprise using millions of dollars worth of computing equipment.

Thus began the department's long, earnest lobbying for more room, an

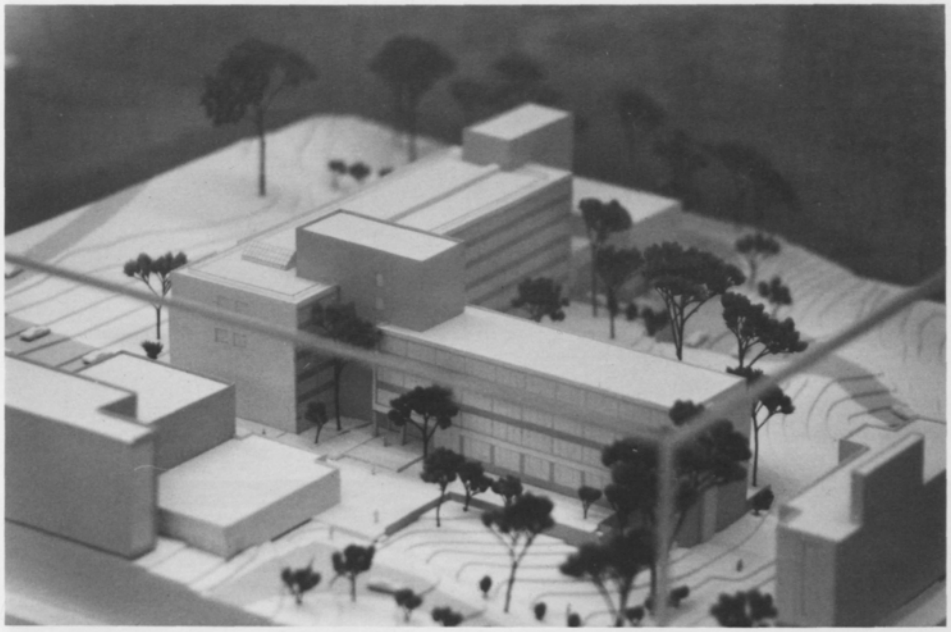


Figure 5. The architects' model showing how Upson Hall will look with the two-story addition. The view is from the northwest (from the direction of Carpenter Hall across the engineering quadrangle). The two floors to be added are on the far wing of Upson.

effort that is finally getting results. This fall construction begins on a $6.8-million addition to Upson Hall for Computer Science that will provide an additional 22,000 net square feet of space (see Figures 4 and 5). Including the third- and fourth-floor space the department now occupies in Upson, Computer Science will have about 37,000 square feet to house its 150 people and its equipment. For the first time, the department will have adequate conference rooms, computing laboratories, administrative office space, and room to house its graduate students. There will be a special robotics laboratory, and a lounge with pantry. The addition will be completely wired for the department's distributed computing facility so that a workstation or terminal connected to the department's central computing faculty can be placed on the desk of every faculty member, staff member, and graduate student. Because of the computing equipment, the addition will be completely air-conditioned.

Meanwhile, the department struggles to maintain a sense of community while being housed in parts of three buildings. Some thirty graduate students and the graduate faculty representative, Charlie Van Loan, are in Carpenter, forty graduate students are in Kimball, and the rest of the department is in Upson. Use of the College's Ethernet, a computer communications network, allows us to have central computing accessible from each location and is one of the reasons this temporary arrangement is feasible.

BUILDING THE CAPABILITIES OF THE DEPARTMENT

Until the late 1970s computer scientists at Cornell were content to use the University's computing facilities. Software projects were in progress, but much of the research was theoretical rather than experimental and just didn't require computers. The department purchased the first computer of its own, a PDP 11/60, in 1977 on a National Science Foundation (NSF) research grant, and that began an era of rapid expansion into the world of experimental computing. One year later, for example, Tim Teitelbaum had finished the first version of his Cornell Program Synthesizer, which provided a much improved programming tool, and Cornell's Computer Services was persuaded to buy enough Teraks so that the new software could be used in course work. As a result, the beginning programming course, CS100, could be taught using a much more effective computing environment.

In 1980 the department was awarded a Coordinated Experimental Research (CER) grant in a new NSF program to equip computer science departments. Our proposal centered around the programming process, since the research of so many of our faculty members in-

Table IV.
THE DEPARTMENT'S COMPUTING FACILITY

Central Facility	Workstations and Microcomputers	Peripheral Equipment
1 PDP 11/60	40 Xerox Dandelions	3 laser printers
2 VAX/750s	9 Suns	3 spinwriters and diablos
2 VAX/780s	4 Symbolics LISP machines	2 line printers
1 Gould/9080	3 HP workstations	1 hard-copy graphics printer
	13 Macintoshes	1 robot arm
	1 IBM PC	1 video recorder

The distributed computing facility of the Department of Computer Science has more CPU units than the entire University had ten years ago. A list of the equipment is given in Table IV.

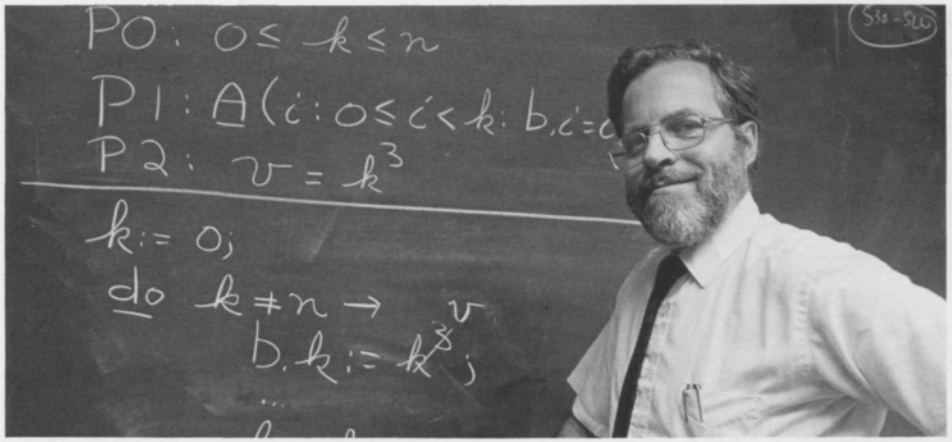
Right: The department's facilities include computing equipment provided through a Coordinated Experimental Research (CER) grant from the National Science Foundation. Terminals are set up in an adjacent room (see the photograph on the inside front cover).

volved programming in one way or another. The grant provided $2.5 million over five years for computing equipment and an operating staff of four. It propelled Cornell into the forefront in a number of experimental research areas, including distributed computing, programming environments, program verification, theorem proving, and robotics.

During the past few years, the department has received several additional NSF equipment grants. The most recent, for $240,000, will be used to acquire a Gould computer that will at least double our UNIX computing cycles. Last year Xerox contributed thirty workstations (called Dandelions) and associated equipment, a gift estimated to be worth about $1 million. Other gifts and grants from AT&T, General Electric, Gould, Hewlett-Packard, and IBM have helped us build a departmental computing facility that now has more raw computing power than Computer Services had for the whole University ten years ago.

Our facility is truly a distributed system, with more than fifty workstations and many more terminals on desks and in public rooms, all connected to our central computers (see Table IV). Everyone, including the administrative staff, uses the facility daily for such operations as word processing and electronic mail service, as well as for research. We anticipate expanding the system so that there is a workstation on every desk and enough "computing cycles" to accommodate the expanding experimental program.

In addition to serving departmental needs, our computing facility functions as the gateway for the entire University to national electronic nets such as the ARPAnet and CSnet. The software and



administration support for this has been provided entirely by our technical staff of four. Frequently we also provide consulting service and assistance to other units of the University that have questions about networking, word processing, operating systems, etc. Last spring, for example, the department worked with Computer Services and the Center for Theory and Simulation in Science and Engineering (the new supercomputing facility, commonly called the Theory Center) to establish needed computer linkages. Also, the University facilities at Langmuir Laboratory were linked to other parts of the campus through our computing facility using PROnet, a token ring system. This year we wrote software that allows the Xerox Dandelions to be integrated into our UNIX environments. Some of this software will be inserted into the new Berkeley 4.3 operating system release and distributed to other universities.

In maintaining, operating, and improving the facilities, an important factor is industrial support. The recently established Industrial Affiliates Program in Computer Science should bring in some of the funds needed for equipment maintenance (this alone costs more than $100,000 a year), as well as for expanding the research program. Corporate members benefit as well as contribute: they have closer contact with our research—joint projects are possible—and with our students.

As we look forward to larger, well-equipped quarters, we foresee the need to expand our personnel as well. We feel we have done very well, with a technical staff of only four, to provide hardware and software support and consulting advice to other groups in the University. But we would like to be able to interact more with units such as the Theory Center and the College of Engineering's Computer-Aided Design Instructional Facility (CADIF). We hope to double the computing facility staff and add more faculty members and research associates.

After twenty years of existence, Cornell's Department of Computer Science is coming of age. The prospects are for continued health and strength in a productive period of maturity.

David Gries, professor of computer science at Cornell, contributes this article in the Quarterly *as chairman of his department. He also collaborated on an accompanying article about his specialty, programming methodology.*

His research interests include related topics such as programming languages and compiler construction. In addition to editing a 1978 book on programming methodology, he has written widely used texts on compiler construction (1971), introductory programming (1973), and the "science" of programming (1981). He is on a committee of the Association for Computing Machinery to review the contents of lower-level computer science courses.

Gries' honors include a Guggenheim fellowship, which enabled him to spend the 1984–85 academic year at Oxford University, England. In 1981 he spent a month in China on the Distinguished Scholar Exchange Program.

He received his Ph.D. in mathematics in Munich, Germany, in 1966, and came to Cornell in 1969 after three years as an assistant professor at Stanford University.

IMMEDIATE COMPUTATION

Or How to Keep a Personal Computer Busy

by Tim Teitelbaum and Thomas Reps

The trip between the keypunch and the computer center was a very familiar route to early computer users. Data prepared in advance were submitted as batches of input to be processed by the central machine.

Timesharing improved matters considerably. The interval between successive runs was reduced from hours to seconds and the trip to the computer center was eliminated. The essence of the batch mode—an alternation between data preparation and program execution—remained dominant, however, even under interactive timesharing.

Today a new way of computing is taking hold. The inexpensive microprocessors of personal computers, more powerful than yesterday's costly mainframes, are bringing about a shift from *batch processing* to *immediate computation*. And the trend appears certain to continue: far more powerful microprocessor workstations are just around the corner.

In the immediate mode of computation, each modification to data has instantaneous effect, since the editing function is embedded within the application program itself. An important result is that a computation is always consistent with the current state of the data, thereby providing useful immediate feedback to the user as the data are manipulated. Extra steps are required, of course, because the data pass through many intermediate states not apparent in the batch mode. These extra steps are acceptable, however, because immediate processing can use surplus processing capacity. (While the unused cycles of timeshared computers are at the disposal of others, the spare cycles of single-user computers are only wasted.)

The trend away from batch processing and toward immediate computation is illustrated by developments in three areas:

- Recent word-processing systems use the immediate-computation paradigm. In the traditional batch-process mode, an input file, consisting of interleaved formatting commands and textual data, is prepared using a conventional text editor. The page layout is created only when this file is submitted to a document compiler. With the newer systems, formatting is performed interactively on a character-by-character basis, and at all times the screen resembles the final page layout. Such editors are called WYSIWYG, because What You See Is What You Get (see Figure 1).
- Electronic spreadsheets, which have become very popular, depend on immediate computation. In such a system a collection of related arithmetic calculations is displayed on the screen and a modification to any of the constants in the formulas causes all affected computations to be updated immediately (see Figure 2).
- In computer programming, "dumb" text editors are being replaced by "smart" editors. In the traditional method, the program prepared with a dumb editor is submitted to a compiler for analysis, error reporting, and code generation. This is unnecessary with smart editors because they exploit language-specific knowledge embedded within them to provide all these services while the program is being edited. Errors are detected early and the delay for compilation is eliminated. As in a WYSIWYG text editor, formatting

"...each modification to data has instantaneous effect."

Figure 1a

```
.pp
This is a right justified paragraph containing
\flitalicized\fR and \fBboldface\fR words. In batch
mode,  it is difficult to tell from the
input file what the final page will look like.
.ip
This is an indented paragraph containing
the formula @x sup 2 - y sup 2@.
```

Figure 1b

This is a right justified paragraph containing *italicized* and **boldface** words. In batch mode, it is difficult to tell from the input file what the final page will look like.

This is an indented paragraph containing the formula x^2-y^2.

Figure 1. A comparison of word processing by a batch method (a) and by an immediate-processing method (b).

The information displayed in a *is used by the batch-oriented word-processing software of the UNIX operating system. This typical input file consists of the interleaved text and commands from which the formatted document of* b *would be produced.*

With a WYSIWYG (What You See Is What You Get) editor, the formatted page, such as the one in b, *is displayed on the screen at all times while the document is being modified.*

Figure 2

Item	UnitPrice	Quantity	Amount	
pen	$0.50	2	$1.00	
pad	$0.75	3	$2.25	
			$3.25	Total

Figure 2. An electronic spreadsheet, an example of immediate computation. The screen is instantly updated in response to each editing revision.

In this example, the Item, UnitPrice, *and* Quantity *columns are user data: the* Amount *column and the* Total *are results of computations. A change in data would immediately be reflected in the computations. For example, if the* Quantity *of pens were changed, the* Amount *and the* Total *would be immediately re-computed and displayed. Entering an improper value such as a nonarithmetic* Quantity *would result in a error message.*

according to program structure is immediate, and as in an electronic spreadsheet, each modification to the program causes all affected analysis, error detection, and code generation to be immediately updated. Our Cornell Program Synthesizer was one of the earliest programming systems to incorporate a smart editor. Since 1979, when it was introduced for instruction in introductory programming, the Synthesizer has become widely recognized as a model for the future development of highly interactive, professional programming environments (see Figure 3).

THE NEED FOR INCREMENTAL ALGORITHMS

Widespread adoption of the immediate mode of computation by new application software is making the study of *incremental algorithms* very important.

Suppose a program computes the function f on the user's data x. If the program follows the immediate-computation paradigm, then the moment the user changes the data from x to x' the program must compute $f(x')$ and discard $f(x)$. Of course, $f(x')$ could be calculated from scratch, but this would usually be too slow to provide instant response. What is needed is an algorithm that reuses as much old information as possible. Because the increment from x to x' is often small, the increment from $f(x)$ to $f(x')$ will also be small, provided that f is continuous. An algorithm that uses information in the old value $f(x)$ to compute the new value $f(x')$ is called *incremental.*

The advantage of an incremental algorithm is illustrated by what happens on the screen when a document is corrected on a word processor that has a WYSIWYG editor. In the WYSIWYG editor, $f(x)$ would be the initial formatted document and $f(x')$ would be the corrected version. Suppose a small change, such as inserting a single character in the middle of a document, is made. It is possible that a major change in the format would result, but this is unlikely for two reasons: *independence* and *quiescence.* The format of the text that precedes the inserted character in no way depends on that character, and is thus unaffected by the change. This is *independence.* In the text that follows the inserted character, the format might change only locally, and even if the inserted character caused some words to snake around to succeeding lines, the propagation of changes would die out and—at least if there is enough space on the last line of the paragraph to prevent the addition of an extra line—the remainder of the document would be unaffected. This is *quiescence.* An incremental formatting algorithm can exploit independence and quiescence so that the minimal amount of reanalysis is done.

We can distinguish between two approaches to incremental algorithms: *selective recomputation* and *differential evaluation.* In selective recomputation, values independent of changed data are never recomputed. Values that are dependent on changed data *are* recomputed, but after each partial result is obtained, the old and new values of that part are compared, and when changes die out, no further recomputations take place. In differential evaluation, rather than recomputing $f(x')$ in terms of the new data x', the old value $f(x)$ is updated by some difference Δf computed as a function of x, x', and $f(x)$.

The spreadsheet example of Figure 2 can be used to illustrate the two approaches.

To illustrate selective recomputation, let us suppose that $\text{UnitPrice}_{\text{pen}}$ and $\text{Quantity}_{\text{pen}}$ are changed to \$1.00 and 1, respectively. Dependency information can be used to determine that $\text{Amount}_{\text{pad}}$ need not be recomputed, since it cannot change. Although $\text{Amount}_{\text{pen}}$ must be recomputed, it turns out to be unchanged, and therefore Total need not be recomputed.

To illustrate differential evaluation, let us suppose that $\text{UnitPrice}_{\text{pen}}$ is changed to \$1.00 and $\text{Quantity}_{\text{pen}}$ is left unchanged. Then the differences

$$\Delta\text{UnitPrice}_{\text{pen}} = \$0.50$$
$$\Delta\text{Amount}_{\text{pen}} = \Delta\text{UnitPrice}_{\text{pen}} \times \text{Quantity}_{\text{pen}} = \$1.00$$
$$\Delta\text{Total} = \Delta\text{Amount}_{\text{pen}} = \$1.00$$

can be computed and used for updating $\text{Amount}_{\text{pen}}$ and Total. Note that with differential evaluation, even if there are hundreds of lines of data, Total can be updated in a single addition.

Figure 3. Three screen images taken from the authors' Pascal program editor to illustrate its immediate error-analysis capability.

The Pascal program in a *shows the names* size, index, list, *and* A *defined in terms of one another. In* b, *after redefinition of* size *from 10 to* x, *an error message appears because the name* x *has nowhere been defined. Changing* x *to 9 instantly removes this error, but introduces another at a distant location, as illustrated in* c.

This editor is, in effect, a spreadsheet for Pascal programs: the computations, updated after each editing transaction and displayed on the screen, concern the structural correctness of the given program.

Photograph below: An initial version of the "smart" editor illustrated in Figure 3 was used for several years at Cornell for instruction in introductory programming.

Figure 3a

```
program p;
const
  size = 10;
type
  index = 1 .. size;
  list = array [index] of integer;
var
  A: list;
begin
  A[10] : = 0
end.
```

Figure 3b

```
program p;
const
  size = x {--NOT A DEFINED CONSTANT NAME};
type
  index = 1 .. size ;
  list = array [index] of integer;
var
  A: list;
begin
  A[10] : = 0
end.
```

Figure 3c

```
program p;
const
  size = 9;
type
  index = 1 .. size;
  list = array [index] of integer;
var
  A: list;
begin
  A[10 {--SUBSCRIPT OUT OF BOUNDS} ] : = 0
end.
```

"Much of the capacity of hardware...is currently going to waste."

SCALING UP THE CORNELL PROGRAM SYNTHESIZER

Sophisticated incremental algorithms were not used in the smart editor of the prototype Cornell Program Synthesizer because exhaustive recomputation was fast enough for small student programs. But before such systems can have a major effect in improving the productivity of software production, they must be scaled up to meet professional requirements. The original Synthesizer whetted appetites, but offered no algorithmic solution to the scaling problem.

In 1981, therefore, the Synthesizer group turned its attention to the fundamental problem of incremental analysis in smart editing environments. In an early paper we proposed the use of a formalism known as *attribute grammars* and showed how incremental analysis could be done in this framework. A major breakthrough occurred when Reps discovered an optimal incremental attribute-updating algorithm. It follows the selective-recomputation paradigm described above, and is optimal in the sense that the amount of processing required in response to a given editing change is linearly proportional to the amount of computed information that changes in value.

ADAPTING SPECIFICATIONS FOR IMMEDIATE COMPUTATION

Traditional systems make use of *imperative programming* languages, in which computation follows an ordered sequence of actions—that is, each action is specified as a function of the previous state. Imperative programming is appropriate for batch-mode computation, in which an input file is processed sequentially, but it is inappropriate for immediate-mode computation, in which data are inserted and deleted in arbitrary order.

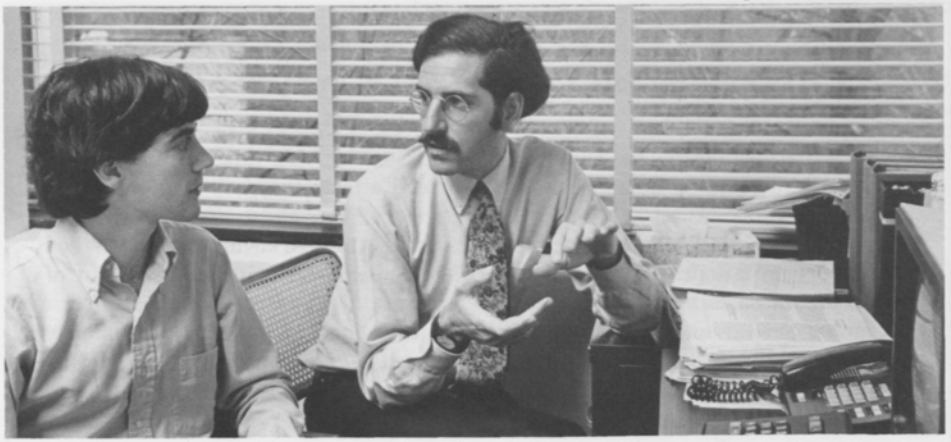
When there is no predetermined order for data and immediate computation is desired, a *declarative specification* is called for. This defines a collection of simultaneous equations whose solution is the desired computation. The salient features of declarative specifications are (1) the order of solution is left unspecified, and (2) the dependence of variables on data and on one another is implicit in the equations. Whenever the data change, an incremental algorithm is used to re-solve the equations, retaining as much as possible of the previous solution.

For example, the "program" executing the spreadsheet of Figure 2 is merely the set of equations:

$UnitPrice_{pen} = \$0.50$
$Quantity_{pen} = 2$
$UnitPrice_{pad} = \$0.75$
$Quantity_{pad} = 3$
$Amount_{pen} = UnitPrice_{pen} \times Quantity_{pen}$
$Amount_{pad} = UnitPrice_{pad} \times Quantity_{pad}$
$Total = Amount_{pen} + Amount_{pad}.$

Changing data is, in effect, changing some of the equations, after which those equations and perhaps other equations must be re-solved.

The attribute-grammar formalism adopted by the Synthesizer group for defining the smart part of program editors is exactly such a declarative specification language. An attribute grammar defines, for each program that a user may create, a corresponding set of simultaneous equations whose sol-



ution expresses the deductions of the smart editor about the given program. Each unknown variable in these equations represents a deduction relevant at a particular point in the program. During editing, each modification to the program causes a related change in the set of equations and their solution. Error messages that appear and disappear on the screen (as in Figure 3) are merely the values of textual variables that change from time to time as the equations are re-solved. The accomplishment of our optimal updating algorithm is that equations arising in the setting of attribute grammars can be incrementally re-solved so that the amount of computation, and therefore the cost, is proportional to the number of variables that actually change value.

The expertise of future smart editors will not be restricted to programming languages. In fact, any application that follows the immediate-computation paradigm is, in effect, a smart editor. Because many of these editors will share similar computational requirements, it is effective to implement *generators* of smart editing systems. Such a generator provides a declarative language which, when compiled together with an incremental updating algorithm, creates the desired application editor.

The Cornell Synthesizer Generator is one such editor generator. Already it has been used to produce editors for a variety of applications: prototype WYSIWYG editors for both text and mathematical formulas, an editor that verifies the correctness of a proof in mathematical logic, program editors for several different programming languages, and even a smart editor for manipulating a database of facts about historic landmarks. A parallel effort, in the program PEOGEN directed by Gregory Johnson of the computer science faculty, has the same goal but is exploring different strategies for providing an interface with the user and for incremental updating. With two groups working in smart-editor generation, Cornell is one of the leading centers for research in this important emerging area of computer science.

Today's powerful stand-alone computers provide virtually free processing. But to make full use of their potential, the impressive advances in hardware must be accompanied by the development of appropriate, innovative software. Much of the capacity of hardware, which can perform millions of operations between every pair of consecutive keystrokes, is currently going to waste. Smart editors know how to use that capacity.

Tim Teitelbaum and Thomas Reps have collaborated in the field of language-based programming systems since 1978.

Teitelbaum is an associate professor of computer science. He has often taught the course CS100, which each year introduces more than twelve hundred Cornell undergraduates to computer programming. He holds the B.S. degree in mathematics from the Massachusetts Institute of Technology and the Ph.D. in computer science from Carnegie-Mellon University. He has been a member of the faculty here since 1973.

Reps received the B.A. degree in applied mathematics from Harvard University and the Ph.D. in computer science from Cornell. His thesis, written under Teitelbaum's direction, received the 1983 Doctoral Dissertation Award of the Association for Computing Machinery. This fall Reps joined the faculty of computer science at the University of Wisconsin as an assistant professor.

A METAPHOR: THE COORDINATED ATTACK PROBLEM

Two allied armies occupy the mountains on opposite sides of a valley, with an enemy army between them. If they mount a coordinated attack, they will win; otherwise, defeat is certain. Alice and Bob, the generals of the allied armies, can only communicate with each other by sending messengers through enemy territory. When messengers are caught, they never divulge the contents of their messages—but of course the messages are not delivered.

Can the generals exchange messages so as to coordinate an attack? Unfortunately for the allies, the answer is no. Alice cannot commit her army to attack after sending the message

m1: "Bob: Attack tomorrow, Alice"

because she cannot be certain that Bob received it. She needs to know whether her messenger got through. However, having Bob reply with an acknowledgment message

m2: "Alice: Received your note, Bob"

doesn't help. This is because even if Alice receives the acknowledgment, she must reason as follows.

> Bob doesn't know whether the messenger carrying his message (m2) got through. Thus, Bob doesn't know if I know whether he got my message. I won't attack unless I am certain that he will attack, and he knows this. Therefore, he must conclude that I won't attack, until he knows that I know that he got my message.

Alice can send an acknowledgment for m2 to Bob:

m3: "Bob: Received your acknowledgment, Alice"

But this still isn't enough. Alice must receive an acknowledgment from Bob for m3, and then Bob needs an acknowledgment for that, and so on. What is required? Alice must know the decision, Bob must know the decision, Alice must know that Bob knows the decision, Bob must know that Alice knows the decision, Alice must know that Bob knows that Alice knows the decision, Bob must know that Alice knows that Bob knows the decision, etc.

To prove that there is no protocol that allows Alice and Bob to coordinate an attack, we can assume there *is* a protocol and derive a contradiction. Suppose Alice and Bob use a protocol that requires the fewest number of messages. Consider the last message sent. Suppose it was sent by Alice to Bob. (The other possibility is symmetric.) Clearly, Alice's decision whether to attack must be independent of the delivery of the last message she sent to Bob. Moreover, Bob's decision must also be independent of whether he received the last message, since the protocol is correct and therefore ensures a coordinated attack even if this message is lost. The last message doesn't affect the behavior of Alice or Bob, so it must be redundant. This, however, contradicts the assumption that the protocol required the fewest number of messages.

REACHING AGREEMENT

A Fundamental Task— Even in Distributed Computer Systems

by Fred B. Schneider, Özalp Babaoğlu, Kenneth P. Birman, and Sam Toueg

Coordinating computers can be as difficult as coordinating the actions of people.

The problem arises in working with a distributed computing system, which consists of a collection of computers interconnected by communication channels. The computers are usually physically separated, and therefore if they fail, they do so independently. This makes it possible for such a system to be fault-tolerant, for data and tasks can be replicated so that if one computer fails, the others can assume its work. Unfortunately, coordinating the actions of a collection of computers when failures *must* be tolerated can be difficult, if not (provably) impossible.

A provably impossible form of coordination is illustrated in the metaphor recounted on the facing page. In the Coordinated Attack Problem, each of the generals can be thought of as a computer and the unreliable messenger as an imperfect communications channel. In terms of the metaphor, we prove that if the communications channel connecting two fault-tolerant computers can lose messages, it is impossible for the computers to agree on whether or not to perform an action suggested by one of them.

Is the Coordinated Attack Problem of practical interest? Unfortunately, it is. If information is to be replicated at computers so that it can remain available despite possible failure of some of those computers, then some arrangement must be made for the replicated data to be kept consistent. This mandates that when one copy of the data is updated, all available copies be updated. Performing the update is an instance of the Coordinated Attack Problem: either all or none of the available copies must be updated.

ATTACKING THE BYZANTINE GENERALS PROBLEM

A second example, the Byzantine Generals Problem, demonstrates another practical problem in distributed computing systems. This problem differs from the Coordinated Attack Problem in two ways. First, Byzantine generals can be traitors and exhibit arbitrary and malicious behavior; in the Coordinated

THE BYZANTINE GENERALS PROBLEM

The generals of the Byzantine army are preparing a final campaign. There are N generals, of whom t, at most, are traitors. The traitorous generals are not known to the others, but may collude and attempt to foil a coordinated attack by the rest. As long as the armies controlled by the nontraitorous generals all attack or all retreat, the campaign will be successful.

The Commanding General, known to all the others, decides whether the army should attack. Generals communicate by means of a reliable messenger service. A protocol is needed that will achieve *Byzantine Agreement:*

Agreement. All nontraitorous generals execute the same action.

Validity. If the Commanding General is not a traitor, then all nontraitorous generals execute her command.

"The generals correspond to processors, traitors...to faulty processors, and the Commanding General's order...to the value of the sensor being read."

Attack Problem, generals always exhibited correct behavior. Second, Byzantine generals have access to reliable communication; in the Coordinated Attack Problem, communication was not reliable.

The Byzantine Generals Problem must be solved whenever processing is replicated in order to counteract the effects of failures in a computing system. This is the basis for triple-modular-redundancy (TMR), which is used by the computing system on board the space shuttle, as well as in other applications in which fault-tolerance is required. The idea is simple. If all processors read the same input from sensors and process it using the same program, then all non-faulty processors will produce identical results. Thus, as long as a majority of the processors are non-faulty, simply voting on the outputs produced by the processors will produce the correct action. A key premise, however, is that all non-faulty processors read the same input. Simply reading from sensors does not ensure that all correct processors will agree on these inputs—a faulty sensor might furnish different values to different processors.

A protocol is required that will permit the non-faulty processors to agree on the value of the sensor. A trivial solution to this problem would be for processors always to use the same value, say 0. This is clearly unacceptable. An agreement protocol must also ensure that if the sensor is non-faulty, processors actually use its value in their computation.

A solution to the Byzantine Generals Problem is exactly the needed protocol. The generals correspond to processors, traitors correspond to faulty processors, and the Commanding General's order corresponds to the value of the sensor being read.

PROCEEDING BEYOND BYZANTIUM

Research aimed at developing and understanding the Coordinated Attack Problem and the Byzantine Generals Problem is actively being pursued at Cornell, M.I.T., and IBM's research laboratory at San Jose. By identifying those aspects of the environment that influence the cost of achieving agreement, researchers learn how to construct agreement protocols that are well suited for particular applications. For example, suppose that the communication channels link only pairs of processors and that up to t processors might be faulty; in this case, achieving agreement can take as many as $t+1$ rounds of message exchange. This is because a faulty processor can send conflicting information along disjoint routes to other processors and can generate spurious messages and then pretend to be forwarding them on behalf of other processors.

One of us (Babaoğlu) has recently shown that as few as two rounds suffice to establish Byzantine Agreement if all processors share broadcast channels. In fact, there exists a continuum of Byzantine Agreement protocols that require between 2 and $t+1$ rounds, depending on the number of broadcast channels available and the interconnection topology of processors. Faster agreement protocols are possible when more processors share more broadcast channels. Clearly, there are trade-offs

involving delay, fault-tolerance, and hardware cost. Broadcast channels help to speed up Byzantine Agreement because when a faulty processor broadcasts a message on such a channel, it has witnesses to its action. Subsequent attempts to send conflicting information can then be detected and ignored by other processors. The Xerox Ethernet and most other local-area communication networks support the requisite broadcast property for these protocols, making them quite practical.

Byzantine Agreement can be made practical even when broadcast channels are not available. It turns out that $t+1$ rounds are necessary to reach agreement only if t failures *actually* occur during execution of the agreement protocol. If f failures occur, where $f<t$, agreement can be achieved in fewer rounds. Thus, the cost of execution is proportional to the amount of fault-tolerance actually needed. Another of us (Toueg) recently was involved in the development of such an early-stopping protocol. It can tolerate as many as $N/3$ faulty processors and terminates in $2f+3$ rounds. Since most executions of the protocol are failure-free, agreement is typically achieved in three rounds. And when t failures occur, the protocol is as efficient as the best previously known protocols—even those that do not support early stopping.

INEXACT AGREEMENTS AND FAULT TOLERANCE

One can devise fast agreement protocols that do not achieve agreement and validity, but come close. For example, an Inexact Agreement (formulated by Schneider) allows processors, each of which has its own value, to converge on some value. While this would not be very useful when the Byzantine generals must decide whether to attack or retreat, it is perfectly appropriate for applications such as clock synchronization. Clocks on computers typically run at different rates and are difficult to synchronize because delays in message delivery are variable—receipt of the message "It is 9:00", for example, tells the receiver very little about what time it is now; it tells only at what time the message was sent. On the other hand, having synchronized clocks can be quite useful, especially in a system intended to be fault-tolerant, since it is easy to detect that a processor has halted by noting that it has taken too long for an expected reply to arrive. An Inexact Agreement can form the basis for a clock synchronization protocol: periodically, processors use Inexact Agreement to agree on a clock value and then reset their local clocks accordingly.

Byzantine Agreement and other protocols that underlie the construction of fault-tolerant systems are complex and subtle. The ISIS Project (under Birman's direction) is concerned with packaging such protocols and building tools that a programmer can use to convert a fault-intolerant program into a fault-tolerant one without worrying about the details of agreement and replication. ISIS programmers have access to a broadcast (agreement) routine that can be used to disseminate information to copies of program modules. The system also provides failure-monitoring facilities so that an ISIS program can reconfigure itself in response to failures. A prototype ISIS is now running on the computer science department's network of DEC VAX

SELECTED READING

Babaoğlu, Ö., and R. Drummond. 1985. Streets of Byzantium: Network Architectures for fast reliable broadcasts. *IEEE Transactions on Software Engineering* SE-11(6): 546–54.

Birman, K. P., T. A. Joseph, T. Raeuchle, and A. El Abbadi. 1985. Implementing fault-tolerant distributed objects. *IEEE Transactions on Software Engineering* SE-11(6): 502–08.

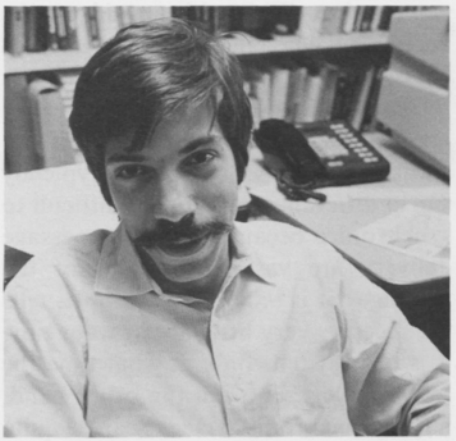
Mahaney, S., and F. B. Schneider. 1985. Inexact agreement: Accuracy, precision, and graceful degradation. In *Proceedings of 4th annual SIGACT-SIGOPS symposium on principles of distributed computing. In press.*

Toueg, S., K. Perry, and T. K. Srikanth. 1985. Fast distributed agreement. In *Proceedings of 4th annual SIGACT-SIGOPS symposium on principles of distributed computing. In press.*

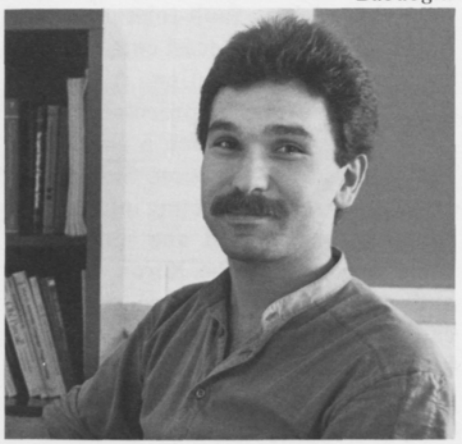
11/780's and SUN Workstations. It has been used to build a number of fault-tolerant application systems and has performed surprisingly well.

Reaching agreements and tolerating faults are modes of conduct for computer-to-computer as well as person-to-person communication. At least in the world of computers, we are finding that logic and imagination are the keys to implementation.

Schneider



Babaoğlu



Birman

Toueg

The four authors of this article are all faculty members of Cornell's Department of Computer Science.

Fred B. Schneider, an associate professor, studied at Cornell for his undergraduate degree, awarded in 1975, and received his doctorate from the State University of New York at Stonybrook in 1978. His main interests are in programming methodology, concurrency, and distributed systems; he is currently writing a text on concurrent programming.

Özalp Babaoğlu, an assistant professor, came to Cornell in 1981 from the Lawrence Berkeley Laboratories, where he was a staff scientist. He holds the B.Sc. degree from George Washington University and the Ph.D., granted in 1981, from the University of California at Berkeley. While in graduate school, he spent a summer at the IBM Research Laboratory in San Jose, and a semester as a foreign scholar at the National Research Council in Pavia, Italy. In 1982 he was a co-recipient of the Sakrison Memorial Award for his contributions to the Berkeley UNIX operating system. His specialties are operating systems, performance evaluation and modeling, and distributed systems.

Kenneth P. Birman received his undergraduate degree from Columbia University in 1978 and his doctorate from the University of California at Berkeley in 1981. At Columbia he helped develop computer systems for analyzing 24-hour electrocardiogram recordings, and after earning his doctorate he spent six months in the cardiology department of the Vienna state hospital in Austria, developing a database system for clinical use. At Berkeley he developed a network version of the UNIX operating system. He joined the Cornell faculty as an assistant professor in 1982.

Sam Toueg, an assistant professor, studied at the Israel Institute of Technology for his B.S. degree, awarded in 1977, and went to Princeton University for graduate work. After receiving the Ph.D. in 1979, he was a postdoctoral fellow at IBM's T. J. Watson Research Center and then came to Cornell in 1981. A specialist in computer networks and distributed computing, he has been a referee for several professional journals.

PROGRAMMING METHODOLOGY

Making a Science Out of an Art

by David Gries and Fred B. Schneider

It doesn't take too long for an intelligent, scientifically oriented person to learn to cobble programs together in FORTRAN, BASIC, or Pascal. Sure, there are mistakes, but everyone makes mistakes, so one simply spends the necessary time debugging. And one gets better at programming simply by doing lots of it. So what do they teach about programming? What is there to it?

This attitude is common and may even be reasonable for casual programming. For any serious programming, however, it invites disaster. A casual program bears little resemblance to the system of a thousand to a million lines of codes that a professional must be able to write (and read) in concert with as many as fifty other people. Such a program must be correct, as simple as possible, and capable of being readily understood, modified, and used by others.

How *does* one write programs that satisfy these requirements? That was the subject of a NATO conference held in Germany in 1968. At the conference the term *software crisis* was often heard, for there was indeed a crisis. The programming industry was being asked to develop larger and more complicated systems of programs, and they didn't have the expertise to do so effectively. The conference led to world-wide recognition that programming was indeed a difficult intellectual activity. The term *software engineering* was coined there to denote the collection of technical and managerial techniques used in the "software life cycle"—in the planning, analysis, design, implementation, testing, documentation, distribution, and maintenance of a programming system—and research in all these aspects began in earnest.

A SCIENCE CONCERNED WITH MENTAL TOOLS

At Cornell, prompted partly by our lack of understanding of how to *teach* programming, we became involved in the study of methods for developing and understanding programs, a field that has become known as *programming methodology*.

Programming methodology has been a central theme in the Cornell department for fifteen years and has influenced our work in other areas. For example, ideas about the process of program development influence thought on compiler construction, programming-language design, structured editors, debugging tools, "pretty printers" (which print a program in an indented format in accordance with the program structure), and computer verification of the correctness of a program or, indeed, of any mathematical proof. These related areas deal with *supplemental* tools used by the programmer; programming methodology in its narrowest sense is more concerned with the *mental* tools that are needed.

Research done so far has convinced us that programming can become a science, based on the knowledge and application of principles, rather than an art, which can be learned simply by watching and doing. We have discovered that programming at its best is a mathematical activity, requiring from the programmer all the taste, elegance, and desire for simplicity that characterizes mathematicians. Our exper-

ience has greatly influenced how we teach programming and how we present algorithms in higher-level courses. So far, most of the research has dealt with small programs, but larger ones are being considered.

In this article we describe some of the basic ideas involved in programming methodology. We use only one small example, but this should be enough to whet your appetite for more. Toward the end we present a couple of problems with the solutions we developed; we encourage you to try to solve them before looking at the solutions.

TAMING COMPLEXITY: THE FIRST NECESSITY

As any programmer will tell you, even a ten-line program can be complex and difficult to understand. Think, then, of the complexity of a ten-thousand-line program! Somehow, the programmer must master the complexity, must prevent it from rearing its ugly head.

The amount of work required to understand a program must be proportional to its length. This will only be the case if the program structure and the interactions between the program segments are kept *simple*. And the longer the program, the more important it is to keep things simple. Computer science already has a branch called *computational complexity*; in contrast, we like to call the field of programming methodology *computational simplicity*.

How do we achieve simplicity? The general method is to introduce suitable notations and use *abstraction*: various aspects of a problem are brought to the fore and others are hidden in the background to be dealt with later. New formalisms are developed, along with notations that allow the expression of concepts and the manipulation of formulas in various ways in order to prove things about them. In essence, mathematics is used, as in any scientific field, to master complexity.

In our research we have turned mostly to formal logic to help us determine what is meant by *correctness* of a program, for without knowing that, it is difficult to write correct programs. This has led to definitions of programming languages in terms of correctness rather than in terms of how a program is executed. And from these mathematical definitions, theories and principles for developing programs have arisen.

This does not mean that every program must be developed and proved correct in a formal manner. It does mean, however, that the programmer with a sound knowledge of the theory and principles behind program correctness and program development can use them in an informal manner, relying on the formalism when it is needed—when the problems become more complex.

WHAT DOES PROGRAM CORRECTNESS MEAN?

A program (or a segment of one) is correct if its execution, begun in any "reasonable" state, ends in a desired final state. That is: if its input variables have proper values, then so will its output variables.

We describe sets of reasonable or desired states by true-false statements, called *assertions*, about the program variables. To illustrate, let us suppose we want a program S to store in an array the cubes of the first n natural numbers, where integer value n is at least 0 and the array is denoted by $b[0..n-1]$. (By convention, if $n = 0$ the array is assumed to be empty.) For example, if we execute the program with $n = 4$, the resulting array will be $b[0] = 0$, $b[1] = 1$, $b[2] = 8$, $b[3] = 27$. Below, we specify S by giving a *precondition* P that describes the set of possible initial states and a *post-condition* R that describes the corresponding final states. In assertion R, the phrase $0 \leq i < n$ means we are interested only in integers at least 0 and less than n; for such integers i, $b[i] = i^3$.

P: $n \geq 0$
R: (**for all** i: $0 \leq i < n$: $b[i] = i^3$)

We say that S is correct with respect to P and R, written as $\{P\}\ S\ \{R\}$, if execution of S begun in a state in which P is true terminates in a state in which R is true. Nothing is said about execution of S begun in a state in which P is not true.

HOW CAN CORRECTNESS BE PROVED?

It is difficult to prove $\{P\}\ S\ \{R\}$ using only our operational understanding of how S is executed. Given some initial state, we can execute the program by hand (or let the computer do it) to determine what the final state is, but to prove correctness using this approach we would have to execute the program once for each possible initial state, and most of us don't have time for that! No, a way must be found that allows us to deduce correctness without relying on the notion of execution, and this calls for a mathematical theory of correctness. For each kind of statement, we need a definition that gives the pairs of pre- and post-conditions related by it.

The theory will tell us, for example, that the following are true about the assignments $x := 0$ and $x := x + 1$ to integer variable x:

$\{0 \bullet y = 0\}\ x := 0\ \{x \bullet y = 0\}$,
$\{x+1 > 0\}\ x := x+1\ \{x > 0\}$.

(Note that $0 \bullet y = 0$ is always true, so that precondition is equivalent to *true*. Similarly, the second precondition is equivalent to $x \geq 0$.)

The possible pre- and post-conditions for a statement should be related by a simple syntactic transformation, and not only by meaning, so that one really can manipulate statements the way one does arithmetic or logical statements. For example, the statement $x := e$, which assigns the value of expression e to variable x, is *defined* by the rule

$\{R^x_e\}\ x := e\ \{R\}$
(for all assertions R).

In this expression R^x_e is the assertion obtained by simultaneously replacing every occurrence of "x" in R by "e". Thus, given that R is to be true after execution of $x := e$, we can determine easily what has to be true before execution: R^x_e. We see that this holds for the two examples given above. For example, in

$\{P\colon 0 \bullet y = 0\}\ x := 0\ \{R\colon x \bullet y = 0\}$,

P is the result of substituting 0 for x in R.

It is rather neat that this simple notion of textual substitution, which is a basic concept of mathematical logic, can be used so simply to define the assignment statement.

Other statements are defined similarly. For example, sequencing of two statements $S0$ and $S1$ is defined:

If $\{P\}\ S0\ \{Q\}\ S1\ \{R\}$,
then $\{P\}\ S0;\ S1\ \{R\}$
(for any assertions P, Q, and R).

This definition allows us to compute the precondition for a sequence of assignments simply by beginning with the postcondition and iteratively working "backward", using the assignment statement definition. For example, it allows us to prove that the sequence $t := x;\ x := y;\ y := t$ exchanges the values of variables x and y. (Below, X and Y denote the final values of x and y, respectively.)

$\{y = X \text{ and } x = Y\}$
$t := x;$
$\{y = X \text{ and } t = Y\}$
$x := y;$
$\{x = X \text{ and } t = Y\}$
$y := t;$
$\{x = X \text{ and } y = Y\}$

Problem 1
COMPUTING CUBES

Write a program to store the cubes of the first n natural numbers in array $b[0..n-1]$. Use only addition operations. (See page 26 for a solution.)

Problem 2
THE MAXIMUM-SUM SEGMENT

Suppose we are given integer array $b[0..n-1]$ for $n \geq 0$. Let $S_{i,j}$ denote the sum of the values of segment $b[i..j-1]$. (If $i = j$, the segment is empty and the sum is 0.) Write a program to store in variable s the largest sum $S_{i,j}$ over all segments $b[i..j-1]$ of array $b[0..n-1]$. (See page 27 for a solution.)

WHAT ABOUT PROGRAM DEVELOPMENT?

Proving a program correct after it has been written is difficult. It makes more sense to develop a program and its correctness proof hand-in-hand, with the *proof* leading the way. When doing this, it is important to write the program specification as precisely as possible (in terms of pre- and post-conditions) because *the specification should drive program development.* To convey this idea through an example, we will consider again the problem of writing a program to store the cubes of the first n natural numbers in array $b[0..n-1]$, with the restriction that since exponentiation and multiplication are expensive, only additive operations should be used in the program. *Try writing a program for PROBLEM 1 yourself before looking at our development on the following page.*

In the event that you have a little trouble, we should point out that SOLUTION 1 was developed using various principles of programming methodology that have only been outlined here. Naturally, you might have difficulty applying them yourself at this point. However, any programmer well versed in the methodology would derive essentially the same program as the one in SOLUTION 1 in perhaps twenty minutes. *How did your solution compare?*

We give one more example without the program development. *Try to develop PROBLEM 2 yourself before reading our solution on a following page.*

The program for PROBLEM 2 has an interesting history. Jon Bentley at Carnegie-Mellon and Bell Laboratories

SOLUTION TO PROBLEM 1

The first step is to specify the program formally by writing pre-and post-conditions:

Precondition P: $0 \leq n$
Postcondition R: (**for all** i: $0 \leq i < n$: $b[i] = i^3$)

Assuming the use of a loop to calculate the elements of array b, our correctness ideas require writing an assertion that indicates what is true of b just before and after each iteration of the loop. To find this assertion, we introduce a fresh variable k (say), which in this case will be what is often called a "loop counter", put suitable bounds on it, and replace n in R by k, yielding the following two assertions $P0$ and $P1$:

$P0$: $0 \leq k \leq n$
$P1$: (**for all** i: $0 \leq i < k$: $b[i] = i^3$)

We can make $P0$ and $P1$ true by setting k to 0. Also, when $k = n$, R is true. And we write the following program:

```
k := 0;
while k ≠ n do begin b[k] := k³;
                     k := k + 1
                end
```

$P0$ and $P1$ are known as *loop invariants,* for they are "invariantly true" before and after each iteration of the loop. One understands the loop in four steps: (0) show that the loop invariants are true just before execution of the loop; (1) show that each iteration leaves them true, so that they are true before and after each iteration and thus upon loop termination; (2) show that the desired result R follows from the loop invariants and the falsity of the loop condition; and (3) show that the loop terminates.

The process used here to argue about correctness should be used to argue about the correctness of *every* nontrivial loop. It is the programmer's task to annotate each nontrivial loop with the necessary loop invariants, because they are a necessary part of understanding the loop. It may seem like a lot of work, but, we maintain, it is simply formalizing what a programmer does anyway when reasoning about why a loop works.

We now have a correct program. However, it uses exponentiation k^3. To get rid of it, we simply introduce a fresh variable x (say) and its definition:

$P2$: $x = k^3$

With this new loop invariant, we can replace the assignment $b[k] := k^3$ by $b[k] := x$. However, $P2$ also depends on k. Before k is increased by 1, x has to be changed to contain $(k + 1)^3$ in order to maintain the truth of $P2$. Since

$$(k + 1)^3 = k^3 + 3 \bullet k^2 + 3 \bullet k + 1$$

and since $x = k^3$ just before each iteration, x can be changed to contain $(k + 1)^3$ by executing the assignment

$x := x + 3 \bullet k^2 + 3 \bullet k + 1$.

This yields the program

```
k := 0; x := 0;
while k ≠ n do begin b[k] := x;
                     x := x + 3•k² + 3•k + 1;
                     k := k + 1
                end
```

The program still contains exponentiation and multiplication. Repeating (twice) the process that introduced x yields the program

```
k := 0; x := 0; y := 1; z := 6;
while k ≠ n do begin b[k] := x;
                     x := x + y;
                     y := y + z;
                     z := z + 6;
                     k := k + 1
                end
```

where there are five loop invariants:

$P0$: $0 \leq k \leq n$
$P1$: (**for all** i: $0 \leq i < k$: $b[i] = i^3$)
$P2$: $x = k^3$
$P3$: $y = 3 \bullet k^2 + 3 \bullet k + 1$
$P4$: $z = 6 \bullet k + 6$

SOLUTION TO PROBLEM 2

The formal specification, using some notation that should be fairly obvious, is

Precondition P: $0 \leq n$
Postcondition R: $s = MAX(i,j: 0 \leq i \leq j \leq n: S_{i,j})$

The program is

```
k := 0; c := 0; s := 0;
while k ≠ n do begin
    c := max(c + b[k], 0);
    s := max(s, c);
    k := k + 1
                end
```

The program is understood in a manner similar to that described for understanding the cube program, using the following loop invariants:

$P0$: $0 \leq k \leq n$
$P1$: $s = MAX(i,j: 0 \leq i \leq j \leq k: S_{i,j})$
$P2$: $c = MAX(i: 0 \leq i \leq k: S_{i,k})$

discovered that a statistician was using a program for this problem that required time proportional to n^3 —the program was actually computing the sums of all segments of the array. Several days later, Bentley returned with an algorithm that required time proportional to n^2, and later one that required time $n \bullet \log(n)$. Another statistician then showed Bentley a program, similar to ours, that required time proportional only to n. During a visit to Cornell, Bentley asked us to write a program for his problem. Several of us who were experienced with the programming methodology came up with the program, independently, in about a half-hour. (Of course, it took time to present it as cleanly as SOLUTION 2 is presented.) We never had to think of the n^2 or $n \bullet \log(n)$ algorithms; the methodology led us quite directly to the solution we have shown here.

WHERE TO LEARN MORE ABOUT PROGRAMMING

Some of the concepts underlying programming methodology have found their way down to undergraduate courses at Cornell such as CS100 and CS211, although in an informal manner, and more will do so as we gain experience and hone our skills. Programming methodology is taught at the upperclass level in course CS400, introduced this spring, as well as at the graduate level; both courses are based on *The Science of Programming*, which was written by Gries in 1981. An additional graduate course, CS613, extends the concepts to deal with *concurrency*, which arises in operating systems, networks, and databases, where various programs are executed simultaneously and communicate through shared data or message-passing. Schneider is writing a text on the subject of this course.

Because of the youth of the field, computer science enjoys the problem that many research results are incorporated quite rapidly into education. Last year's research problem has moved into this year's first-year graduate course and will be in next year's senior-level course. Our students, even in the early undergraduate years, are learning about a relatively new discipine as it develops.

The practicality of theoretical research is demonstrated every day not only in advanced computing centers, but in university classrooms where tomorrow's practitioners are learning their skills.

David Gries is chairman of the Department of Computer Science and a specialist in programming methodology (see the biographical sketch on page 11).

Fred B. Schneider is an associate professor in the department. A specialist in concurrent programming, operating systems, and distributed systems, he has also written another article in this issue in collaboration with three of his colleagues (see pages 18–22). He is on the College Board Committee on Advanced Placement in Computer Science, which prepares an examination that reflects much of what is taught at Cornell in the introductory courses.

ROBOTICS AND COMPUTER SCIENCE

by Dean B. Krafft

Robotics as a significant area of computer science is a recent phenomenon. In the traditional view, robotics is the study of designing, controlling, and instrumenting robots, and clearly this view characterizes a large body of problems that must be solved to use robots effectively. A robot must be able to move quickly and precisely to a location and perform some action such as grasping an object; this is a problem in design and control. Next, there must be some mechanism for storing a sequence of robot actions for later playback; this is a control problem. Finally, the most sophisticated current robots have the ability to modify their actions in response to sensory input; this presents a problem in instrumentation and control.

For research in these three essential areas—design, control, and instrumentation—a traditional engineering approach seems appropriate. But in very different ways, computer science can also contribute to robotics.

Let us consider what robots do. They manipulate physical objects in accordance with a set of prespecified instructions and their perceptions of the workspace. This differs from computer-controlled hard automation, long the province of mechanical, electrical, and industrial engineers, in that general-purpose, programmable robots are not specialized to solve a single physical problem. Instead, they can potentially perform a number of arbitrary physical manipulations.

One way in which robots can be used is simply to replace hard automation. By manually stepping them through the series of motions they are to perform, they can be programmed for a single task. The advantage is that when the task changes, it is not necessary to scrap the production line, but only to reprogram the robots for the new task. This is a rather limited use for a very powerful tool, however.

GENERALIZING ROBOTICS USING COMPUTER SCIENCE

By making use of off-line programming, robot actions can be specified without actually moving the robot through the steps, and this allows the system to provide more powerful capabilities and to deal with much more complex situations. For example, numerical codes can be generated by a computer using information from a computer-aided-design database and used for controlling machining or assembly operations. A robot can make use of sensory feedback and a database of possible actions to guide its behavior; if it detects errors, it can take alternative actions and proceed with the task. Off-line programming also allows the robot to deal with variations in the objects or processes it is handling, without having had all the cases preprogrammed by hand.

Off-line programming brings with it new requirements, of course. Programmers can no longer make use of the actual robot and physical prototypes to answer all their questions about position, action, and movement. Instead, the robot task must be described in terms of computer models of the workspace and the objects to be manipulated.

The next step immediately suggests itself. If the robot system has models of the physical objects with which it is dealing, why not use these models to

"...off-line programming...allows the system to provide more powerful capabilities and to deal with much more complex situations."

simulate and reason about the objects? This would allow robot programmers to work at a much higher level, saving time and money. For example, such reasoning might allow the robot system to figure out a grip, to plan motion, to determine the stability of an assembly (and thus how to move it from one location to another), and to understand the behavior of nonrigid objects (such as a pair of pliers it must pick up).

We have now arrived at a much more general view of robotics. Functions involving high-level programming, reasoning about objects, and perception are qualitatively different from the design, control, and instrumentation tasks described earlier. The problems and goals are much less clearly defined and the techniques required to elucidate and solve them are different. It is in dealing with these areas that the analytic tools of computer science can make great contributions.

USING THE ANALYTIC TOOLS OF COMPUTER SCIENCE

In approaching problems in robotics, computer scientists follow a general sequence of steps. They analyze a problem to identify its components; abstract the problem to eliminate unneeded specifics; restate the problem in a formal and rigorous manner; and (ideally) provide provably correct solutions to the formal problem statements. Then they try to apply the solutions to real problems, determine errors arising from failures in understanding the original problem, and repeat the cycle. In solving such problems, researchers must make use of tools and techniques for complexity reduction, formal reasoning, modeling, data abstraction, algorithmic analysis, and so forth. Computer science provides the training and tools to solve large, ill-defined, complex problems, and computer scientists can apply this training and these tools to the problems of robotics.

The process has already begun. Computer scientists are already studying the design of robot languages and are working on such problems as how to plan the motion of robots through potentially crowded workspaces and how to program robot arms to grip objects. Researchers in the area of artificial intelligence have been studying certain aspects of robotics for years; they are working on how computer vision can be used to recognize arbitrary scenes, and how mobile robots can be made capable of navigating through unknown terrain. Research undertaken so far has been limited, however, to specific subproblems of robotics.

SOLID MODELING FOR RESEARCH IN ROBOTICS

The first step in applying computer science to robotics is to develop models of objects. One requirement is to represent physical surfaces and volumes; over the last five years, a number of systems, called solid modelers, have been built to do this kind of modeling. These systems represent objects as mathematical equations pertaining to either the surfaces of the object or a set of primitive solids, and they represent surfaces as either planar polygons or quadrics (a quadric is a surface that can be represented with a second-degree polynomial).

Although much progress has been

An example of solid modeling development at Cornell is this representation of the loggia of the University's planned center for the performing arts. The modeling was done by Phil Brock, a graduate student working in the Program of Computer Graphics. A view of the adjacent corridor is shown in color on the front cover.

made in solid modeling, much more work remains to be done. As an example, let us look at a current problem. Most machined objects can be almost entirely represented by quadric surfaces. However, the blending and filleting surfaces that smooth off sharp inside and outside edges of machined objects tend to be much more complex, not easily represented in the solid-modeling systems now available. A designer may spend far more time describing the blending surfaces of an object, even though the exact shape of these may not be essential, than in specifying the functional surfaces. The computer science robotics research group at Cornell, led by John E. Hopcroft, has recently developed a technique for automatically deriving fourth-degree blending surfaces for joining quadric-surface solids. With this system, a designer needs to specify only the functional surfaces and the system will fill in the blending surfaces. This technique has been used in the design of blending surfaces for several assemblies, including a gate valve (see the image reproduced in color on the front cover; the blending surfaces are blue).

Another interesting problem in solid modeling involves the rapid display of complex solid models. In designing a physical assembly, graphically displaying the partially created object is an important part of the user/system interaction. When the user makes a change in the assembly, it should not be necessary to wait for minutes or hours to see a display of the new object. At the same time, the more accurate and complete the display of the object, the more useful it will be to the designer. The work of two groups at Cornell typifies research on this problem. One team, led by Donald P. Greenberg, director of the Program of Computer Graphics, is working on the fast display of very complex environments for design. Objects are represented by polygonal panels or by procedurally represented surfaces, including quadratic surfaces and swept splines. The other group, led by Hopcroft, is attempting to display solids whose surfaces are represented by implicit algebraic equations. Polynomials up to fourth degree or even greater are used.

There is a trade-off between the techniques involved: Polygonal panels are simple to display, but an accurate representation of a curved surface requires a large number of panels. On

the other hand, curved surfaces usually can be modeled with just a few higher-degree algebraic surfaces, but the individual surfaces are much harder to render. For the fastest display, some combination of techniques will be needed.

REPRESENTING ASSEMBLIES: THE NEXT STEP

One of the most important challenges in object representation is to advance beyond present-day solid modelers. Existing solid-modeling systems can represent single solids, such as the result of a machining or casting operation, quite well, but they have the drawback of being able to represent assemblies only as the fixed relative positioning of a group of simple solids. There is no capability to represent joints (and their degrees of freedom), attachments, flexibility, and so forth.

A simple extension to existing solid-modeling systems provides a few of these extended capabilities by using feature-based models to represent assemblies. This method involves the identification of a set of sites as being the focus of some activity, such as attachment or insertion. These sites are explicitly represented in the object model, and actions affecting the object are specified in terms of them.

Along these lines, the Cornell computer science robotics group has developed a system that uses a feature-based model to describe and perform some simple assemblies. The current version of the system deals with a very restricted assembly environment—it builds Lincoln Log cabins. The user provides what is effectively an exploded diagram of the desired assembly (Figure 1) and the system, running in Interlisp on a Xerox Dandelion (an $8,000 workstation), converts this diagram to actual VAL code (Figure 2). This code then drives our PUMA 560 robot to perform the assembly. The conversion process involves planning grips, motions, and transformations on an internal computer model of the actual workspace (see Figure 3). While the existing problem domain is still quite restricted, the underlying structure of the system should be able to handle more complicated and realistic assembly problems.

The lack of any representation for joints, attachments, and so forth in existing solid modelers has other implications. For one thing, it means that solid modelers are totally inadequate to determine the physical behavior of an assembly, and this leads to the next important area of object representation: reasoning about objects. Given a representation for a collection of objects, a designer needs to be able to make predictions about their real-world behavior. The questions can be as simple as whether or not an object will fall over or fall apart under the force of gravity, or they can be as complex as how long a car will last if it is driven over a cobblestone road.

Recently Rick Palmer, a Cornell graduate student in computer science, developed an algorithm for testing the stability of a collection of arbitrary objects. Unfortunately, his work also showed that the general problem is very costly to solve (it falls into the class of problems that computer scientists term NP-complete). There is reason to believe, however, that the problem is much more tractable for structures that can be built without temporary scaffolding, and the robotics group plans to use this idea as the basis for developing a new system for planning and performing assemblies. One of the problems in automating assembly procedures is to determine the stability of intermediate objects; the Cornell group intends to develop a system that uses heuristics to choose paths that appear likely to lead to stable partial assemblies. The stability algorithm could then be used to test the proposed assembly steps. In cases in which the intermediates are not stable, it might be possible to automatically deduce the scaffolding necessary to render the partial assembly feasible.

REPRESENTING TASKS: A FURTHER CAPABILITY

The next step, beyond reasoning about simple object behavior, involves representing and reasoning about tasks. Two simple ways to represent tasks illustrate some of the challenges. One way is to simply represent a task as a sequence of actions; this is the method used in all existing robot-programming languages

A system that can be used in describing and performing simple robot assemblies is being developed by the Cornell computer science robotics group. These figures illustrate a prototype version that has been devised for building a simple assembly—a Lincoln Log cabin.

Figure 1. An exploded diagram description of a Lincoln Log assembly. The letters label the logs.

Figure 2. The VAL code for assembly, used to drive the robot. This was produced by the computer from the exploded diagram of the desired assembly (Figure 1).

Figure 3. A sample execution of the assembly-planning system. This is part of the process of converting the diagram to the VAL code.

Right: Krafft demonstrates the assembly of a Lincoln Log cabin.

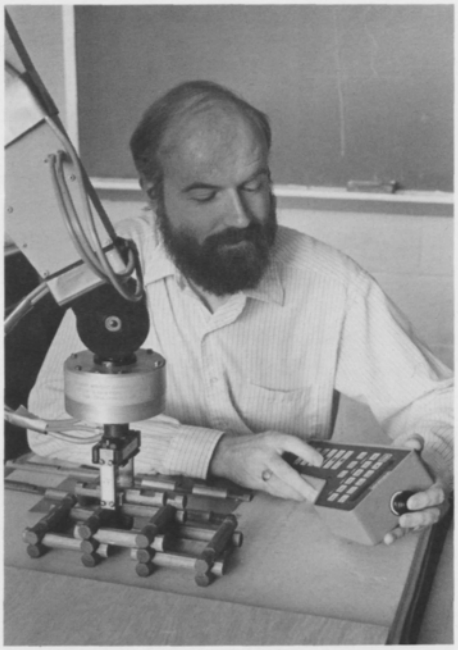


Figure 1

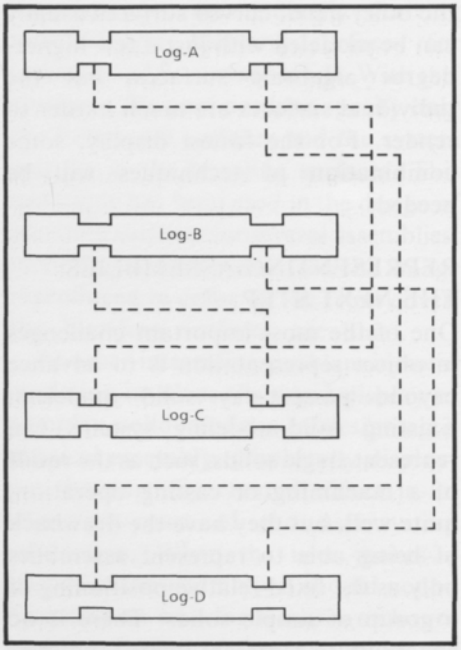


Figure 2

```
-> (build)
(Setting up modules)
(Initializing variables)
(Compiling assembly graph)
(Assembly Planner called to compile)
(Ordered list is:)
        (transformation-node-type1 transformation-node-type2
        transformation-node-type3 transformation-node-type4)
(Transformation Manager called to plan:)place
(Grip Planner called to grip object:)"log1 origin"(at location:)
        {444090.0 557340.0 -356630.0 1.0}
(Grip Planner called to plan an ungrip)
(Motion Planner called to move robot object:)"manipulator"
(Motion Planner called to move robot object:)"manipulator"
(Transformation Planner called to plan:)logstack
(Grip Planner called to grip object:)"log2 upper left slot"(at location:)
        {444090.0 557340.0 -356630.0 1.0}
(Grip Planner called to plan an ungrip)
(Motion Planner called to move robot object:)"manipulator"
(Motion Planner called to move robot object:)"manipulator"
(Generating VAL code in file VALCODE.PG)
```

Figure 3

```
val.example        Thu Sep 26 16:23:17 1985        Page 1,line 1

.PROGRAM VALCODE
MOVE TRANS (400.000000, 0.000000, 300.000000, 90.000000, 90.0 0.0)
MOVE TRANS (444.090000, 557.340000, -16.630000, 0.000000, 90.0 0.0)
MOVE TRANS (444.090000, 557.340000, -316.630000, 0.000000, 90.0 0.0)
DELAY .1
OPENI
MOVE TRANS (444.090000, 557.340000, -356.630000, 0.000000, 90.0 0.0)
DELAY .1
CLOSEI
MOVE TRANS (444.090000, 557.340000, -316.630000, 0.000000, 90.0 0.0)
MOVE TRANS (444.090000, 557.340000, -16.630000, 0.000000, 90.0 0.0)
MOVE TRANS (600.000000, 100.000000, -19.340000, 0.000000, 90.0 0.0)
MOVE TRANS (600.000000, 100.000000, -319.340000, 0.000000, 90.0 0.0)
SPEED 20
MOVE TRANS (600.000000, 100.000000, -359.340000, 0.000000, 90.0 0.0)
DELAY .1
OPENI
.END
```

(see Figure 2 for an example). With this method it is easy to specify and perform the task, but it may be hard to reason about the effect of the task on the workspace. A second technique is to represent a task as a set of objects in two states, the current state and the desired state. Here the effect of the task is specified and therefore simple to reason about, but determining the sequence of actions may be a formidable problem in reasoning.

The second of these representation methods has obvious applications in robotics. It encompasses the classic problems of how to move a robot arm and a gripped object through a workspace—problems that are generally extremely difficult, as verified by theoretical work at Cornell and elsewhere. The method also applies to compliant motion (the motion of objects in contact), which is of critical importance in many robotics problems. Except for a few specialized applications, such as spray-painting or laser cutting, almost all uses of robots involve compliant motion. An important theoretical result was obtained recently by Gordon Wilfong, who was then a graduate student in the department, working with Hopcroft. They proved that if it is possible for two objects to move from one configuration in which they are in contact to another configuration in which they are in contact, then there is a motion that continuously maintains contact between the objects.

Work in the areas of motion planning, compliant motion, and related task-reasoning problems is just beginning to break through from the purely theoretical to results that can be applied to real problems in robotics.

THE POTENTIAL IMPACT ON INDUSTRIAL PRODUCTIVITY

As the use of robots grows and as they are applied to more and more sophisticated tasks, the need for computer science research in robotics will increase dramatically. A broad area of research based on the problems of robotics offers a significant challenge to computer scientists; if it can be met, computer science will make a valuable contribution to a number of areas of production and engineering.

Research of this kind has already begun, but if continued progress is to be made, it is essential that scientists at Cornell and other institutions recognize its significance. The prospect is that over the next decade, robotics will become one of the most vitally important research areas in all of computer science.

Dean B. Krafft has been a research associate in Cornell's Department of Computer Science since he received his Ph.D. here in 1981. As an undergraduate, he majored in mathematics at Carleton College.

Krafft's specialty is robotics, the subject of this article, and he has been heavily involved in Cornell's robotics program, which is funded by the National Science Foundation. He is an author, with his Cornell colleague John E. Hopcroft, of a more extensive paper, "The Challenge of Robotics for Computer Science," which will appear in Advances in Robotics, *vol. 1:* Algorithmic and Geometric Aspects of Robotics, *to be published by Lawrence Erlbaum Associates.*

"As the use of robots grows ...the need for computer science research in robotics will increase dramatically."

COMPUTER ARCHITECTURE

The Software–Hardware Interface

by Jon A. Solworth

The architects of computers, like those who plan buildings, must design for soundness of construction, effective use of materials, economy, and function. And like their counterparts in the building industry, computer architects make use of the available technology.

With computers the technology today is based on semiconductors and is the result of combined efforts of specialists in such fields as electrical engineering, semiconductor physics, chemistry, and materials science. Computer architects must effectively combine this technology with structure to attain high-performance systems. This involves the added element of software, for the architecture–software interface plays a significant role in determining the structure of advanced computer systems, and will become dominant as the size and sophistication of software continues to increase. It is here that computer science disciplines are critical to computer architecture.

Some of the software is general-purpose and functions to make the computer usable. Compilers, for example, allow programmers to write in languages like FORTRAN, BASIC, or Pascal; operating systems manage resources such as processors, memory, disk drives, and terminals. Other software is specific to the applications that run on the machine. To achieve high performance—speed, reliability, and ease of programming—the interactions among the architecture, the system software, and the applications programs must be taken into account.

Figure 1

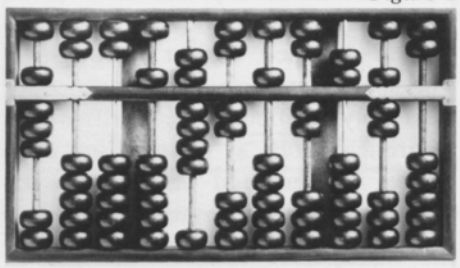


Figure 1. The bead abacus, an Asian form of one of the earliest kinds of calculators. (Photo courtesy of IBM.)

THE RISE IN CAPABILITY AND ITS LIMITATIONS

The challenges to computer architecture have changed and rapidly accelerated as the modern computer has developed. Indeed, one of the striking differences between traditional architecture and computer architecture is the speed of change: The introduction of steel as a building material in the 1900s—making possible the skyscraper—was the last major change in the underlying technology available to the architects of buildings. In computer architecture, three major technological innovations—vacuum tubes, transistors, and now integrated circuits—have been introduced since 1946, and each has caused or will bring about large shifts in the ways computers are built.

The history of the computer dates back to the use of devices for counting, such as the abacus and the sixteenth-century calculator. With these instruments the operations were sequenced (entered) by hand. In the nineteenth century, the Jacquard loom was invented, and although it was not a calculating device, it introduced a new sequencing method—the use of punched cards to order the steps necessary to weave a complex textile. Some time

Figure 2. Napier's Bones, a sixteenth-century precursor of the slide rule. When the vertical pieces are arranged to show a given number across the top row, a multiple can be read from the line corresponding to the desired multiplier. (Photo courtesy of IBM.) In the example at right, the selected "bones" correspond to 2698. Readings are made from right to left, as in ordinary addition, with numbers that appear in the parallelograms added together and two-digit sums carried over. For instance, 2698 multiplied by 3 yields:

0 (6+1) (8+2) (7+2) 4 or 0 8 0 9 4. Thus 2698 x 3 = 8094.

Figure 3. Jacquard's loom, a nineteenth-century invention that introduced the use of punched cards. The cards automatically controlled the sequence of operations to produce the desired pattern in the woven fabric. (Photo courtesy of IBM.)

Figure 4. Part of the Babbage Differential Engine, the first mechanical computer. This late-nineteenth-century machine was steam-driven. (Photo courtesy of the Science Museum of London.)

Figure 2

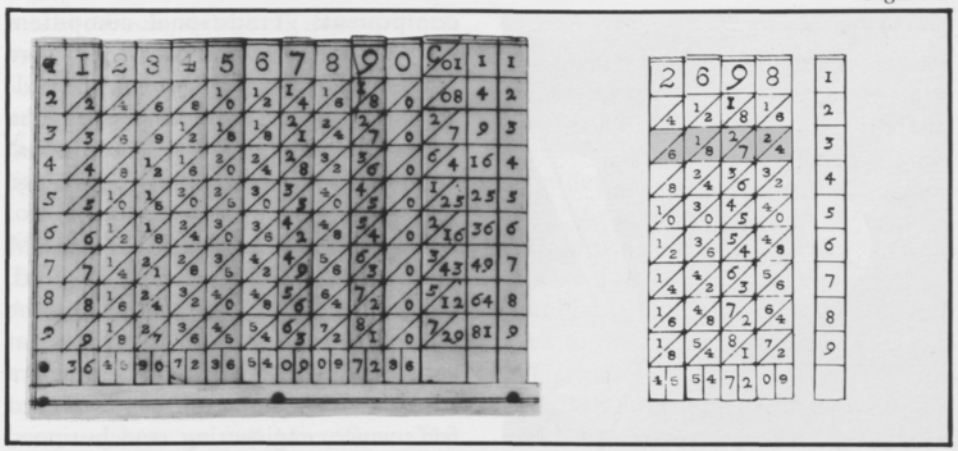


Figure 3

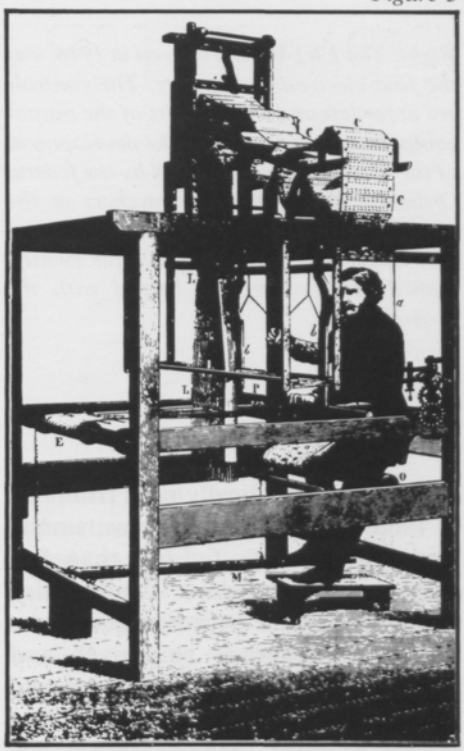
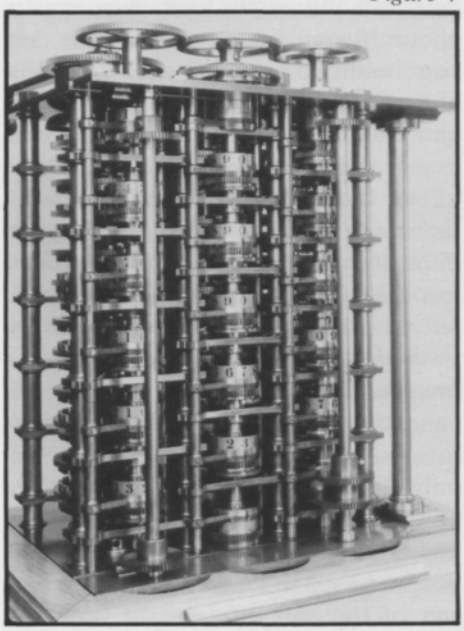


Figure 4

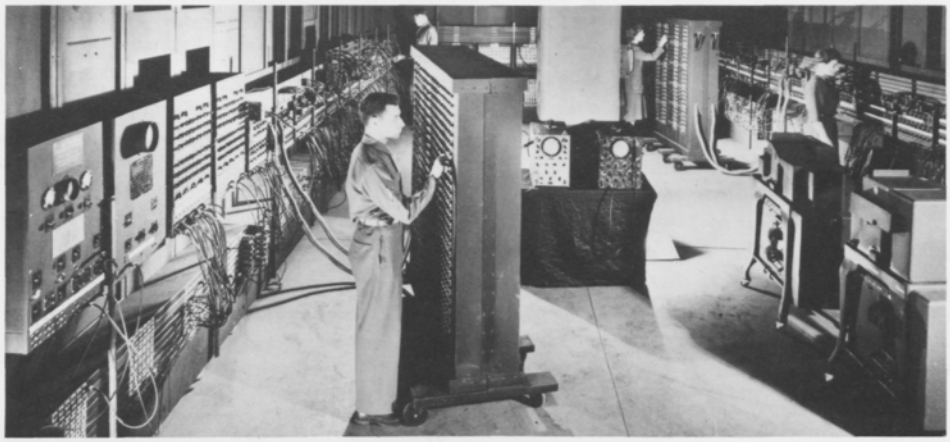


after that, Charles Babbage set out to build the first computer, the completely mechanical, steam-driven Differential Engine. Because Babbage's invention preceded the introduction of interchangeable mechanical parts, his attempt to build a working computer failed; yet he pioneered many of the concepts used in computers even today.

The first electronic computer, the ENIAC, was completed in 1946. It used vacuum-tube technology and was very slow by today's standards. Because of their expense, not a great many vacuum tubes could be used (the ENIAC had eighteen thousand). The number was limited also by the relatively high failure rate: increasing the number of tubes meant a higher probability of failure and therefore a decrease in the amount of computing actually accomplished.

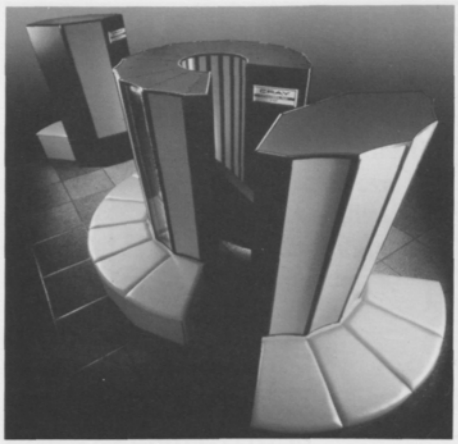
From the time of the ENIAC to the present, computers have increased in speed by five orders of magnitude. Now computers use transistors fabricated on integrated circuits (chips) instead of vaccum tubes. The number of transistors that can fit on an integrated

Right: The ENIAC, introduced in 1946, was the first electronic computer. The controls are at far left and a small part of the output equipment is seen at right. The development of the machine was sponsored by the federal Ordnance Department; researchers at the Moore School of Electrical Engineering at the University of Pennsylvania (which loaned the photograph) helped with the programming.



circuit has dramatically risen from one in 1960 to about 250,000 now; such a chip typically sells for less than five dollars. Making chips smaller also makes them faster because signals have a shorter distance to travel and switching time is reduced. The gains of the last twenty-five years cannot go on forever, however; fundamental physical laws will ultimately limit chip size to about 10 million transistors, and this size limitation (coupled with problems of heat generation) imposes a limitation on the speed with which we can practically compute.

The most important physical limitation is the speed of light. It may be difficult to believe that the speed of light can really limit the size of computers, but it must be remembered that in state-of-the-art computers the times involved are very short, of the order of nanoseconds, and in one nanosecond (one billionth of a second) light travels only about 4 inches. The CRAY-XMP, a modern supercomputer, performs an addition in 18 nanoseconds, but the time required to send a signal from one end of the computer to the other (a distance of 56 inches) is 14 nanoseconds. A fast computer must be small or it will spend more time communicating than computing. Size reduction brings problems of heat dissipation, however, so there is a tradeoff.



TAKING UP THE CHALLENGE IN UNIVERSITY RESEARCH

Since it is becoming increasingly difficult to make computers faster by improving the underlying technology, an alternative is being attempted: the use of more, rather than faster, components. Traditional computers have several adders; what if there were a thousand? Potentially, a thousand-fold speed-up in computation would be possible. A difficulty is that traditional computer architectures cannot manage so many resources because the control issues become dominant: instead of performing computations at full speed, adders would be idle, waiting for the control mechanism to decide what should be done next. If computers are to achieve the computation rates needed for science, engineering, and business, new and perhaps radical computer architectures must be discovered. The foremost challenge is the effective use of massive parallelism for solving computationally extensive tasks.

Within this context, the primary focus of university research is to combine thousands to millions of processors so as to provide aggregate execution rates of over a billion instructions per second (GIPS). The raw potential of such computing ensembles eclipses the capability of today's largest computers, but this increase in performance will not come free—new

"The most important physical limitation is the speed of light."

methods of designing computers, systems software, and applications software are required. It is only through an integrated approach on these three fronts that the potential of massively parallel computers can be exploited.

MICROFLOW: AN APPROACH TO PARALLEL COMPUTING

At Cornell the aim of a project on which my colleague Alexandru Nicolau and I are working is to combine systems software with architecture in one coherent system we call Microflow. The most important piece of software, from this perspective, is the compiler. In the Microflow method, classical compiler technology is used to analyze the properties of applications programs, yielding information that enables the computer to reduce the time it takes for execution.

Even with traditional architectures, this technique speeds up computing by a factor of 2 to 5—an impressive speedup that occurs without involving the programmer at all. But while compiler-implemented program optimization is an important part of preparing computationally intensive jobs for traditional architectures, it is absolutely imperative for massively parallel programs. Not only is the speedup potentially much larger, but there are more details to deal with—the program must know where the data are and on what processor a particular computation is being run. The compiler's analysis simplifies the code, and provides feedback to the user as well.

To get high performance from a massively parallel computer, it is necessary to rewrite critical programs so that they will run efficiently. To help users do this, we are working on a parallel programming environment that combines highly efficient, hand-tailored code for critical sections along with the automated analysis and optimization features of the compiler. This environment is being targeted not only at the Microflow architecture, but also at several computers being acquired by the supercomputer center recently established at Cornell by the National Science Foundation.

The systems software we envision for Microflow will enable us to build a computer that would otherwise be impractical to program. Microflow is aimed at exploiting very fine-grain parallelism. This means we expect to take advantage of parallelism not at the level of running separate programs on the same machine, or manually separated parts of the program in parallel, but at the level of *interleaving* instructions (for example, additions) in the same part of the program. To efficiently accomplish this, we must reduce the control overhead that is inherent in traditional design. One way to do this is to move the overhead to the compiler so that it is handled only once—when the program is compiled, and not when the program is executed. A second way is to introduce novel architectural mechanisms that will efficiently execute the code produced by the compiler and environment. We are working on both these techniques.

ACCESSING AND ENHANCING MEMORY

Along more traditional lines, Nicolau and Kevin Karplus, a member of the electrical engineering faculty, are in-

vestigating methods to speed up von Neumann architectures, which are the basis of all commercially produced general-purpose computers.

A problem is the so-called von Neumann bottleneck—the time it takes to get information from the central memory to the processor. This bottleneck is alleviated, in part, by retrieving more information from memory than is needed, with the hope that the extra information will be used shortly by the processor. This technique works well when the accesses are predictable, but when they are not the processor is idled while it waits for memory. Another way to reduce the bottleneck is to improve the prediction of accesses to memory so that requests can be anticipated better. A Cornell project called ROPE (Ring of Prefetch Elements) is exploring novel programming techniques intended to ensure that the processor very rarely has to wait for access to memory in order to receive instructions.

Another important problem in computer architecture is support for special programming languages and applications. The goal of Cornell research in this area is to improve the performance of very sophisticated programming languages or, equivalently, to reduce the time penalty associated with their use. One approach is to build very-high-performance systems out of unconventional languages; another is to provide environmentally rich computer systems that will enable programmers to build software more quickly.

A popular programming language structure (called a *class*) is Simula, which incorporates both the data and the operations performed on data in a single entity. My research group is investigating an enhanced-memory architecture called a Negotiator, which should implement such systems better than traditional architectures can.

COMPUTER-AIDED DESIGN: KEY TO ARCHITECTURE

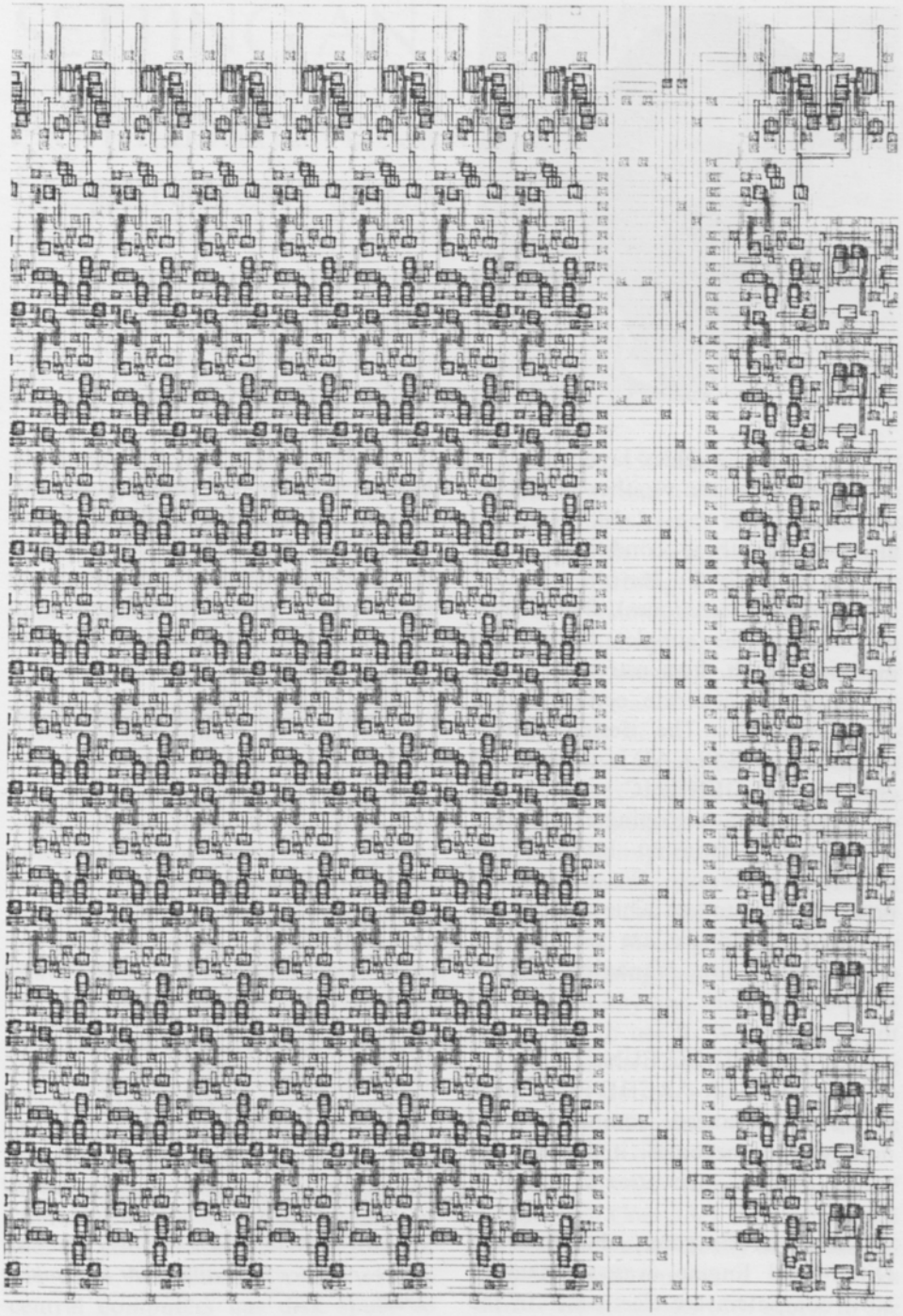
Computer-aided design (CAD) of integrated circuits is a key technique in computer architecture. Several of our projects require prototype computers for testing theories, and without sophisticated CAD, these prototypes would be too complex for us to build.

Errors in design result in unusable chips. There is no way to patch a chip; the design must be changed and the whole manufacturing process repeated from scratch. Not only is this time-consuming and expensive, requiring typically several months and costing several thousand dollars, but there is a worse problem. A chip containing, say, 250,000 transistors would have about a hundred pins where electrical signals are input or output, and there is no way to probe the interior (as there is in conventional electronic design) to identify the error.

All sophisticated chips produced today, either in industry or in a university research laboratory, are both simulated and validated. *Simulation* applies signals to a computer model of the input/output (I/O) pins and produces the same output as the actual chip would. Since the simulation is a computer model, it is possible to probe interior points and diagnose problems. Simulation is dynamic: it tests that the chip will work on certain supplied inputs. Because the designer has to be responsible for making sure that the tests are complete, simulation serves as an assurance test rather than a guarantee that the chip is correct. *Validation*, on the other hand, is an absolute check on certain correctness properties—for example, it verifies that wires have at least some minimum width. Validation is always preferable to simulation, but may not be available for all properties.

Validation of timing can also be accomplished with the help of CAD. The transistors on an integrated circuit can be thought of as implementing a series of switches that propagate electrical signals along wires, much as a railroad switching yard routes trains on tracks. A problem in designing a chip, then, is to ensure that valid information is always on the wires when these are used in a computation; in the railroad analogy, this corresponds to ensuring that trains do not collide in the yard. At Cornell, Karplus is working on graph theoretic algorithms to analyze and verify that integrated circuits meet this important condition.

A third pertinent area of CAD is the management of the design process and effective exploration of design alternatives. With today's VLSI (very-large-



This computer plot of a section of a VLSI layout was generated in Solworth's laboratory in three colors. The section shows a queue of nine bit elements, eleven elements deep—part of a switch for a large parallel-processor communication network. The structural appearance of the tiny chip produced from this design will be almost exactly like this plot.

scale integration) chips, it is difficult to evaluate all the design alternatives; CAD simplifies the job.

A GENERIC APPROACH TO PROBLEMS OF LAYOUT

An area of particular interest to Cornell researchers is *layout*, the placement of components on an integrated chip. An integrated circuit contains transistors and wires embedded in a plane; the database for a million-transistor chip, for example, contains about five million geometric shapes. These components should be packed as tightly as possible without overlapping, but since the design begins with an electrical circuit rather than with a planar mapping, the embedding, or layout, must be worked out later to achieve maximum efficiency and compactness.

Layout is now performed manually at a color graphics workstation, a method that is a linear extrapolation of the historical process of integrated-circuit (IC) design. This method is reaching the limits of its ability to handle effectively the complexity of current chips, for the designer, in effect, must draw each object in the integrated circuit. To the extent that objects are

reused, they can be replicated rather than redrawn, but there is a tradeoff: the more general-purpose (and hence more easily replicated) the objects, the lower the performance of the chip.

In a project called GENERIC (for GENERation of Integrated Circuits), my research group is working on improved layout techniques. Since programming languages have been used for thirty years for both abstraction and specification of computation in a wide variety of problems, we believe that a language approach would be suitable for IC design. The starting point is a description, in GENERIC, of the electrical circuit that implements the desired function. Using both library routines and special-purpose routines, the designer creates an embodiment of an integrated circuit by "fleshing out" the original circuit.

In explaining GENERIC, we often draw an analogy between designing an integrated circuit and flying an airplane. The graphics approach is like flying by sight; it is simple, develops intuition about how to "fly", and is the least expensive method. However, flying by sight (normal graphics layout) is not sufficient when the weather is bad or when the plane must land at a busy airport. For these conditions, it is necessary to fly by instruments (use programming techniques). Before they can replace the simpler fly-by-sight method, though, these instruments must be sufficiently accurate, complete, and comprehensible. The design instrumentation of VLSI CAD is not yet good enough, and GENERIC seeks to improve it. We are developing instruments for assessing (without looking) the quality of the design. We are creating powerful operators to achieve high-level objectives. And we are working on mechanisms for managing design details.

THE CONTINUING FUNCTION OF COMPUTER ARCHITECTURE

However advanced computer technology is or becomes, it depends on actual physical structures in order to function.

The first computer, the Babbage Differential Engine, was mechanical. Now computers are electronic. In the future they may be based on optical or biological technology. But regardless of the available materials, techniques, and underlying technologies, computers must be designed as overall structures that work. Architecture will continue to be an important aspect of the remarkable evolution of the computer.

Jon A. Solworth, a specialist in computer architecture, VLSI, and computer-aided design, joined the Cornell computer science faculty in 1984. He holds two degrees in computer science from New York University and soon will receive his doctorate.

While he was an undergraduate, Solworth worked as a programmer and analyst for the Standard Security Life Insurance Company. As a graduate student he served as a research assistant and assistant research scientist at New York University and also had experience as a consultant to Bankers Trust.

SETTING AN EXAMPLE

Administrative Computing in Cornell's Department of Computer Science

by Diane Duke and Michele Fish

Our department acquired its first computer, a DEC PDP 11/60, in 1978. The faculty members may have thought of computers as tools for research and instruction, but the administrative staff soon developed other ideas! It wasn't long before an interactive terminal was placed in the main office for shared use by the staff members. The department's exciting—but sometimes harrowing—move into the use of computers for administrative purposes had begun.

At the time, the department had no computer technicians or staff programmers; it was the faculty members and graduate students who had to keep the hardware running and the software usable. After a while, many procedures such as *booting* the system after *crashes* and taking system *dumps* became routine. (The system crashes when, for an unknown reason, it simply stops working; one then reboots the system by loading it into memory from magnetic tape. Dumping means to store the contents of disks on magnetic tape so that the information can be recovered if the disks fail. Currently, one of our central computers has disk space to hold almost 550 million characters of information, and losing them would be catastrophic.) A staff member became familiar with these procedures and began performing them; and for the next four years, many problems that arose with the system were fixed by an office staff member. It was not the typical "secretarial" work.

Since our departmental computing equipment is mainly for experimental research, this was the main consideration in deciding on an operating system. Less importance was placed on word-processing capability. For this reason, the system is not as "user friendly" as those generally used for word processing, but it provides much more flexibility and function for those who learn to use it. We learned a lot about computer systems. And, having been innovators, we have been able to train people in other departments as well as in our own.

Today all the administrative staff members in Computer Science, as well as everyone on the faculty and technical support staff, have on their desks interactive terminals with access to the main computers. Typewriters are rarely used. Management, administrative, and clerical duties are performed with the same computer system used for research.

COMMUNICATION USING ELECTRONIC MAIL

How does the computer system affect the day-by-day conduct of department business? Take communication as an example. No one has to check a mailbox for telephone messages because they are all delivered by electronic mail, along with many intradepartmental communications. If a call for x cannot be received personally, for whatever reason, the receptionist simply types "mail x", along with the message, into the terminal; the time and date are automatically recorded. Within seconds, the message is stored in x's files and appears on his or her terminal screen; an answer can be sent at a convenient time. There is also a broadcast capability so that we can alert a group of people to a special problem, an imminent meeting, or some other matter of common concern.

The electronic mail service is not limited to the department, for the system is connected to several national and international networks such as the ARPAnet of the Department of Defense, and CSnet, which serves the computer-science community. Depending on the network used, transmission of a message can take from five minutes to one day. An impressive demonstration of the usefulness of network communication was provided recently by the chairman of our department, who used it to write a twenty-page report in collaboration with colleagues in Seattle, Atlanta, and Palo Alto. A draft was mailed electronically to the authors in round-robin fashion, and within two days each had read and edited it on the computer.

Besides our own address file, we have access to central address files (one in Boston and one in Palo Alto) listing people around the country who are in computer science or related fields. One can query these from a desk terminal and get both a post-office address and an electronic address in a few seconds. Some day such electronic address files will be available for other disciplines as well.

More and more units at Cornell are acquiring their own computers and a University-wide network is being installed. Already the College of Engineering has its own Ethernet that links the computers in all its buildings. Soon it will be possible to mail electronically an article like this to *Engineering: Cornell Quarterly*, and an edited draft could be returned the same way. An initial difficulty is that different units have different word-processing or computer equipment and software; ways will have to be found to translate from one system to another.

HANDLING CORRESPONDENCE AND OTHER CLERICAL TASKS

In our office most correspondence is typed using the computer system instead of a typewriter. A letter can be composed on the system, mailed electronically to others for comments or revision, and checked electronically for spelling errors. Figures and tables can be incorporated easily by picking them up from other files. The final step is to produce the *hard copy*—the actual letter—using either a typewriter-like device or a laser printer. When almost-identical but personalized letters are to be sent to a number of people, they can be prepared by "filling in the blanks" in a standard format, thus avoiding having to type each one separately; this procedure is used, for example, by the graduate faculty representative and his secretary in corresponding with prospective graduate students.

Many other clerical tasks are easily accomplished using the system. Technical reports, annual reports, mailing labels, address lists, course-enrollment adds/drops—all these are handled routinely.

The preparation of our annual report is an example of how the system can reduce work. The report contains, for each faculty member, lists of University activities, professional activities, lectures given, and publications. A secretary mails each faculty member the section of the previous report that pertains to him or her, and the faculty member (or a secretary) changes it electronically to bring it up to date. A simple electronic collation completes the work on that part of the annual report. The information can also be used to update the faculty member's vita, which is kept in a central file.

The preparation of research proposals is greatly facilitated by the computer system because many parts that are required—technical data, vitae, references, budgets, etc.—can be gathered from stored files. From the administrative point of view, the computer system is particularly useful in preparing the proposed budget for a research project: various packages, such as spreadsheets, are very helpful. Once a project is funded, the budget can be transferred to active accounting records within the system.

Accounting is another big job that is made easier by the computer system. Records from Cornell's computer must be verified and interpreted at the departmental level, and it is simple to check charges to an account against the original approved budget on file in our system. We can also verify available balances in research accounts and prepare financial reports to the principal investigators. Department records

Duke and Fish

pertaining to such items as purchase and work orders, accounts payable/receivable, equipment inventory, and the payroll are also kept on the system. Spreadsheet programs are utilized in maintaining and handling the data. For security reasons, some personnel and management records have not yet been computerized, but soon we will get around this problem by using personal computers and workstations that make it possible to store sensitive data locally and still have access to the central system.

Student records, which are especially complicated for an intercollege department like Computer Science, are well handled by the interactive system. An Apple Macintosh database package has been adapted for keeping the records of undergraduate computer science majors in both the College of Engineering and the College of Arts and Sciences. Computerized records for graduate students, including information about admission, financial aid, and academic status, makes life easier for the graduate faculty representative and also for his secretary.

A SYSTEM FOR EVERYONE IN THE DEPARTMENT

We know from personal experience that the interactive computer system has increased the effectiveness of the department's administrative staff. One of us, Diane Duke, works with personnel and accounting records, and the other, Michele Fish, coordinates the undergraduate academic program. Other staff members use the system in carrying out their special responsibilities. Anita Affeldt works with the graduate faculty representative in administering the graduate program. Geri Pinkham uses the department computer to keep accounting records. Donette Isenbarger has attended a seminar on administration using the department's new Xerox workstations and is now responsible for training the rest of the clerical staff on the Xerox STAR system.

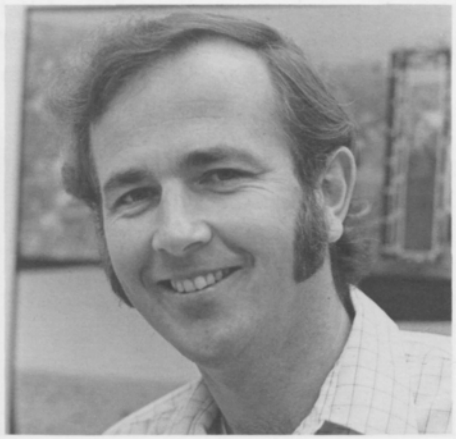
These staffers and many others in the department are using the continually changing and growing computer system, or are learning to do so. They have discovered a new truth—that the requirements for administrators and secretaries have expanded in a very short time as terminals have replaced typewriters. The office, as well as the classroom and the research laboratory, is at the forefront of advances in computer science.

Diane Duke, administrative manager of the Department of Computer Science, came to Cornell nine years ago as an administrative aide. She now supervises a staff of twelve, including research and department secretaries, an accounts coordinator, administrative aides, and office assistants. She majored in business education at Oregon State University.

Michele Fish joined the department as a secretary in 1980 and was one of the first staff members to become proficient in the use of computers. Currently an administrative aide, she coordinates the department's supervision of undergraduate majors in two colleges. She is a participant in the University's employee degree program and recently enrolled in the College of Human Ecology with a major in human development and family studies.

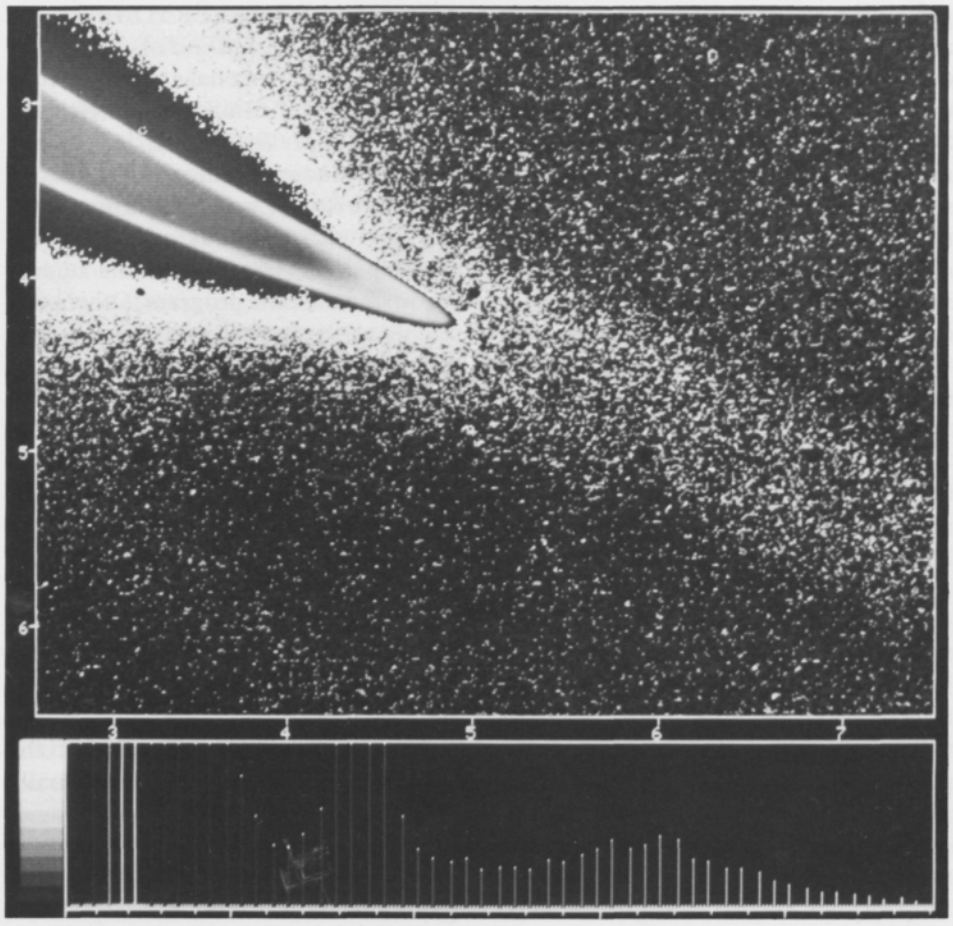
VANTAGE

Burns



■ An additional ring around Jupiter, discovered by researchers including a Cornell engineering professor, will be an unexpected feature for NASA's Galileo space probe to observe when it explores the giant planet in 1988–1990. In fact, the discovery has brought about a rerouting of Galileo to avoid passage through the ring.

The engineering professor is *Joseph A. Burns* of the Department of Theoretical and Applied Mechanics, whose specialty is celestial mechanics. He worked with Mark R. Showalter, a Cornell postdoctoral associate in astronomy who was a graduate student at the time of the study, and two NASA scientists, Jeffrey N. Cuzzi and James B. Pollack. They found evidence of the previously overlooked ring by reprocessing data of a particular Voyager 2 image.

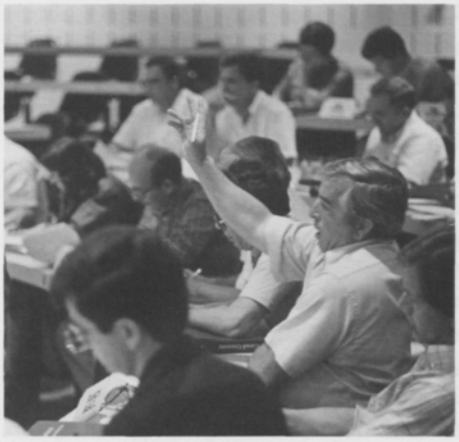


Right: This reprocessed image from Voyager 2 shows the newly discovered "gossamer" ring as a faint band at center-right. The bright ring and the halo ring are at upper left.

Particles detected so far in the band—called the "gossamer" ring because of its faintness—are in the micrometer range, about the size of smoke particles. NASA's concern about Galileo is that even particles as small as this might penetrate the probe's heat shield, causing heat "spikes" that on entry into the planet's atmosphere could shatter the heat shield, jeopardizing the mission. Individual ring particles have lifetimes that are very limited because of the ring's location in the planet's Van Allen belt, a region of intense electromagnetic radiation. Presumably, the ring is continually regenerated by micrometeoroid collisions into larger unseen parent bodies. The particles, dark red in color, may be continually coated by the outpourings of the volcanoes on the Jovian moon Io.

The gossamer ring extends some 210,000 kilometers from the planet. It is about twenty times as faint as the previously known "bright" ring, which can be seen only by the largest telescopes on Earth. The other known feature is a "halo" ring that extends about 10 degrees out of the main ring plane. The Galileo mission, scheduled for launch next spring, is expected to provide images and detailed data about these features, as well as Jupiter's satellites, surface, and atmosphere, during the probe's voyage of two or more years through the Jovian system.

■ Key problems for United States manufacturers were discussed at a week-long conference this summer that brought more than forty senior executives to the campus. "Managing the Next Generation of Manufacturing Technology," offered for the second year, was held August 12 through 16.



Above: C. Reid Rundell, an executive vice president at General Motors, gave an after-dinner talk about changes in production and process design that are being implemented in the manufacture of the new Saturn automobile, a world-class car in terms of quantity and cost.

Other after-dinner speakers were Al Zettlemoier of IBM, who gave the keynote address, and Cornell Professor Alfred E. Kahn, who talked on the economic future—government policy, international competition, and business decisions.

Left: Industrial delegates participated in the discussions.

The main topics were applicable engineering developments, business analysis of technology, and factors involved in introducing change. John A. Muckstadt of the College of Engineering, who is director of the Cornell Manufacturing Engineering and Productivity Program (COMEPP), and L. Joseph Thomas of the Johnson Graduate School of Management were the symposium directors. Other faculty participants represented University units in architecture, in arts and sciences, and in industrial and labor relations. Speakers from the engineering college and their areas of expertise included Muckstadt (distribution systems), Herbert H. Johnson (new materials), K. K. Wang (CAD/CAM/CAE or computer-aided design, manufacturing, and engineering), Noel MacDonald (automated manufacture of integrated circuits), and John E. Hopcroft (robotics).

REGISTER

■ Six engineering schools or departments opened the fall term with new faculty members.

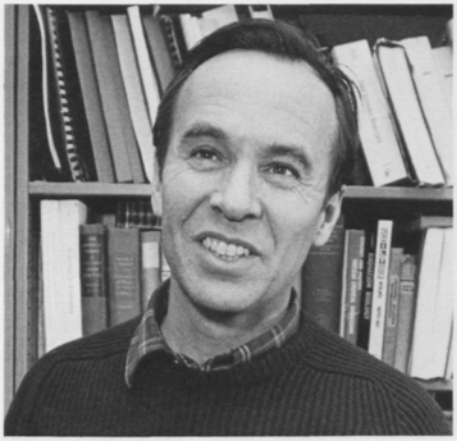
Harry E. Stewart, a specialist in geotechnical engineering, joined the School of Civil and Environmental Engineering as an assistant professor in the Department of Structural Engineering. He received the B.S. degree in chemistry at the State University of New York (SUNY) at Brockport, and then switched to civil engineering, earning the B.S. at SUNY Buffao and the M.S. and Ph.D. at the University of Massachusetts at Amherst. After receiving the doctorate in 1982, he taught at Amherst and at the University of South Carolina before coming to Cornell. Stewart's honors include membership in Tau Beta Pi, Chi Epsilon, and Sigma Xi. He belongs to the American Society of Civil Engineers, the Transportation Research Council, and the United States branch of the International Society of Soil Mechanics and Foundation Engineering.

A new associate professor in the Department of Computer Science is *Dexter C. Kozen*, who came to the University from IBM Research Laboratories, where he was a research staff member and manager of the theory of computation group. After graduating in 1974 from Dartmouth College (where he received the John G. Kemeny Prize in Computing), he came to Cornell for graduate work and received his doctorate in 1977. Before joining IBM, he was a postdoctoral fellow at the University of California, and he has been a visiting professor at Aarhus University in Denmark and an adjunct professor at Columbia University. He is a member of the Association for Computing Machinery and the American Mathematical Society.

Victor Solo, formerly an associate professor at Harvard University, joined the School of Electrical Engineering as an associate professor. A specialist in stochastic processes, he has also served as an assistant research scientist at the University of Wisconsin and as a visiting associate professor at Purdue University. Solo earned two undergraduate degrees in Australia: the B.Sc. in mathematics at the University of Queensland (where he won the mathematics prize) in 1971; and the B.Sc. in statistics, with first class honors, at the University of New South Wales in 1973. The following year he received the B.E. degree in mechanical engineering, with first class honors, from the University of New South Wales, and in 1979 he was awarded the Ph.D. in statistics by the Australian National University. He spent two years with the Commonwealth Scientific and Industrial Research Organization as a consulting statistician. He is an editor or reviewer for several journals in engineering and statistics, and belongs to the Institute of Electrical and Electronics Engineers and the Institute of Mathematical Statistics.

A new assistant professor in the School of Mechanical and Aerospace Engineering is *Michel Y. Louge*, whose field is thermal sciences. Louge received the degree of Ingenieur from the École Centrale des Arts et Manufactures in 1978, and took his graduate work in mechanical engineering at Stanford University, which awarded him the M.S. in 1979 and the Ph.D. in 1985. Since the fall of 1984 he had been a

Cady



Hammer

process development engineer at Shell International.

In the School of Operations Research and Industrial Engineering, *William J. Cook* has joined the faculty as an assistant professor. Cook came to Cornell from West Germany, where he was an Alexander von Humboldt Research Fellow at the University of Bonn for two years. His degrees are the B.A. from Rutgers University, the M.S. from Stanford University, and the Ph.D. from the University of Waterloo. His specialty fields are combinatorics and combinatorial optimization.

Also new to the operations research and industrial engineering faculty is *Michael Phelan*, who studied for his M.S. and Ph.D. degrees at Cornell. He earned the B.A. degree at the State University of New York college at Oneonta in 1978. He has had experience in applications and systems programming with Townsend and Greenspan, Inc. His professional interests include inference problems from stochastic processes, particularly counting processes.

Timothy J. Healey, a specialist in solid mechanics, is an assistant professor in the Department of Theoretical and Applied Mechanics. His three degrees in civil engineering are the B.S. from the University of Missouri and the M.S. and Ph.D. from the University of Illinois at Champaign-Urbana. At Illinois he received the 1985 Chester P. Siess Award, which recognizes outstanding doctoral work and promise for research. His experience includes two years with Agbabian Associates of Los Angeles beginning in 1978, and teaching mathematics at the University of Maryland last year. Healey is a member of the honorary societies Chi Epsilon, Tau Beta Pi, and Omicron Delta Kappa, and belongs to the American Mathematical Society, the American Academy of Mechanics, and the Society for Industrial and Applied Mathematics.

■ A new administrative appointment at the College of Engineering is that of *K. Bingham Cady* as associate dean for college affairs. Cady, a professor of nuclear science and engineering and of applied and engineering physics, has been serving as associate dean for professional programs. Cady has been at Cornell since he received his doctorate from the Massachusetts Institute of Technology in 1962. In addition to his academic and administrative work at the University, he has has extensive experience as a consultant to industrial firms and national facilities, including the Knolls Atomic Power Laboratory, the U.S. Atomic Energy Commission, Brookhaven National Laboratory, Hanford Engineering Development Laboratory, and Fauske and Associates. He has held a Ford Foundation Preinduction Scholarship, a Bethlehem Steel Industrial Fellowship, a Woodrow Wilson Fellowship, and a U.S. Atomic Energy Commission Fellowship in Nuclear Science and Engineering. He is a member of Phi Eta Sigma, Sigma Xi, and Tau Beta Pi.

■ The new director of the Laboratory of Plasma Studies is *David A. Hammer*, professor of nuclear science and engineering and a specialist in plasma physics, nuclear fusion, and high-power electron- and ion-beam physics. He succeeds *Ravindra N. Sudan*, who is now deputy director of the Center for

Shoemaker

Liu

Theory and Simulation in Science and Engineering, one of four recently established National Advanced Computing Centers. Hammer came to Cornell in 1977 after seven years as a research physicist with the Naval Research Laboratory, and has been on the faculties of the University of Maryland, the University of California at Los Angeles, and Imperial College, London. He is a fellow of the American Physical Society, a senior member of the Institute of Electrical and Electronics Engineers, and a member of the American Association for the Advancement of Science and of Sigma Xi.

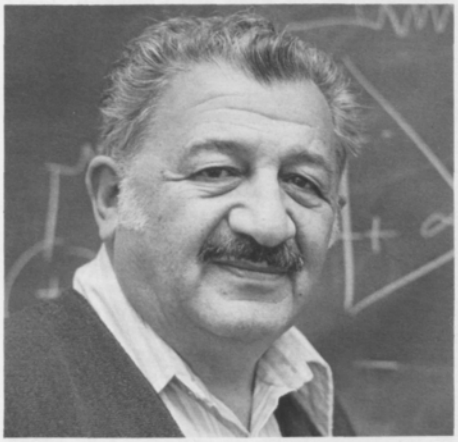
■ In administrative changes at the School of Civil and Environmental Engineering, Professor *Christine A. Shoemaker* has been named chairman of the Department of Environmental Engineering and Professor *Philip L.-F. Liu* has succeeded her as associate director of the School.

Shoemaker has been at Cornell since 1971, when she received the Ph.D. degree in mathematics from the University of Southern California. After a year as a research associate here, she joined the environmental engineering faculty. A specialist in pest management, water resources systems, and mathematical ecology, Shoemaker is a member of several graduate fields at Cornell: applied mathematics, agricultural engineering, entomology, and ecology, in addition to civil and environmental engineering. She has served on National Academy of Sciences panels on pest management and on groundwater contamination. She is a member of the U.S. National Committee for the Scientific Council on Problems in the Environment—a standing subcommittee of the Environmental Studies Board of the National Academy of Sciences. Also, she is on an advisory panel on pest management for the Food and Agriculture Organization of the United Nations. She is a member of the Entomological Society of America, the American Geophysical Union, the Biometric Society, and the Operations Research Society of America.

Liu, a specialist in fluid mechanics and coastal engineering, is a professor in the Department of Environmental Engineering. He joined the Cornell faculty in 1974 after earning the B.S. degree at National Taiwan University, and the M.S. and Sc.D. at the Massachusetts Institute of Technology. He has also been a visitor at the University of Delaware and the California Institute of Technology. His honors have included the 1978 Walter L. Huber Research Prize awarded by the American Society of Civil Engineers (ASCE), a Justice Foundation faculty fellowship at Cornell, an Engineering Foundation fellowship in 1980, and a Guggenheim fellowship in 1980. He is a member of the ASCE and the American Geophysical Union.

■ *Benjamin Nichols*, professor of electrical engineering, has been named associate director of the School of Electrical Engineering. A Cornell faculty member since 1946, Nichols has had a continuing interest in educational policies and techniques; his work in this area at the College of Engineering has included directing the former Division of Basic Studies and serving on the committee that set up the current core

Nichols



Scheele

curriculum. Nichols is a Cornell alumnus and holds the Ph.D. in geophysics from the University of Alaska.

■ At the National Research and Resource Facility for Submicron Structures (NRRFSS), *Gregory Galvin*, a postdoctoral associate, and *J. Peter Krusius*, an associate professor, have been named associate directors. Galvin works with the user program, which accommodates researchers from across the nation who come to work at the national facility, and Krusius is in charge of the Cornell Program on Submicrometer Structures (PROSUS), a program for industrial affiliates.

■ Recently appointed staff members who work with students include *Judy Jackson*, coordinator for advising in the Office of Admissions, Advising, and Records; and *Linda Van Ness*, coordinator of the Engineering Cooperative Program.

Jackson holds a B.A. degree in French from the University of North Carolina at Greensboro and an M.A. in Francophone African literature, geography, and foreign policy from Buckell University. She has taught at Susquehanna University and headed the Office of Minority and Foreign Student Advising at Bucknell.

Van Ness has been at Cornell since 1981, most recently as a staff member in the dean's office. She holds an associate degree in psychology from Genesee Community College.

■ *George F. Scheele*, associate professor of chemical engineering, has been named a fellow of the American Institute of Chemical Engineers. He was cited for his contributions to engineering education and to research in fluid mechanics.

A graduate of Princeton University and a Ph.D. from the University of Illinois, Scheele has been at Cornell since 1962. He has also been a visiting professor at the University of California at Berkeley, a "Year-in-Industry Professor" at E. I. duPont de Nemours, and a research engineering with the Dow Chemical Company. He is currently serving as associate director of the School of Chemical Engineering.

■ The National Best Student Section of the Society of Women Engineers is our own SWE chapter. The selection of the Cornell group, announced early this term, was made in recognition of the chapter's ambitious and well-rounded program in 1984–85. The award is accompanied by a cash grant from the Union Carbide Corporation.

SWE's events during the year included an awards banquet, a visit to Corning Glass Works, a barbeque, an ice-skating party, and a party on the engineering quadrangle open to everyone at Cornell. The society also sponsored programs for prospective students: High School Day in November, and Accepted Candidates Weekend in April. A program in career guidance included seminars, workshops, and the publication of members' resumes. Younger members were helped academically and personally through tutoring and a Big/Little Sister program.

The 1984–85 president was Margot Haartz, a senior in mechanical engineering. This year's president, Margaret Au, is an electrical engineering major.

FACULTY PUBLICATIONS

Current research activities at the Cornell University College of Engineering are represented by the following publications and conference papers that appeared or were presented during the four-month period February through May, 1985. (Earlier entries omitted from previous Quarterly *listings are included here with the year of publication in parentheses.) The names of Cornell personnel are in italics.*

■ AGRICULTURAL ENGINEERING

Steenhuis, T. S., C. Jackson, S. K. J. Kung, and *W. Brutsaert.* 1985. Measurement of groundwater recharge on eastern Long Island. *Journal of Hydrology* 79:145–69.

Taylor, R. W., and *G. E. Rehkugler.* 1985. Development of a system for automated detection of apple bruises. Paper read at conference, Society of Manufacturing Engineers, 25–28 February 1985, in Chicago, IL.

■ APPLIED AND ENGINEERING PHYSICS

Del Priore, L. V., and *A. Lewis.* 1985. Vanadate, tungstate, and molybdate activate rod outer segment phosphodiesterase in the dark. *Biochimica et Biophysica Acta* 845:81–85.

Dowben, P. A., *D. Mueller, T. N. Rhodin,* and *Y. Sakisaka.* 1985. Molecular bromine adsorption and dissociation on iron and nickel surfaces. *Surface Science* 155:567–83.

Dowben, P. A., *Y. Sakisaka,* and *T. N. Rhodin.* 1985. Angle-resolved photoemission from bromine chemisorbed on Ni(100). *Journal of Vacuum Science and Technology* A3:1855–59.

Park, S., L. C. Rathbun, and *T. N. Rhodin.* 1985. Effect of inert ion bombardment on chemisorption and etching of aluminum films in Cl_2, Br_2, CCl_4, and CBr_4. *Journal of Vacuum Science and Technology* A3:791–94.

Rhodin, T. N., M.-H. Tsai, and R. V. Kasowski. 1985. Chemisorptive bonding of carbon monoxide on Ni(001) and Fe(110). *Applied Surface Science* 22/23:426–43.

Sakisaka, Y., T. Rhodin, and *D. Mueller.* 1985. Angle-resolved photoemission from Fe(110): Determination of E(k). *Solid State Communications* 53:793–99.

■ CHEMICAL ENGINEERING

Anderson, C. C., and *F. Rodriguez.* (1984.) Polymeric plasticizers for poly(methyl methacrylate). In *Proceedings, American Chemical Society Conference on Polymeric Materials Science and Engineering,* vol. 51, pp. 609–13. Washington, DC: ACS.

Calado, J. C. G., and *P. Clancy.* (1984.) The development of a flexible intermolecular potential energy function for ethylene. *Revista Portuguesa de Quimica* 26:85–91.

Calado, J. C. G. 1985*a*. The dialogue between order and disorder. *Boletim da Sociedade Portuguesa de Quimica,* 2d ser., 20:9–10.

———. 1985*b*. The need for new developments in thermodynamics. Paper read at ChemPor 85, 4th International Chemical Engineering Conference, 15–19 April 1985, in Coimbra, Portugal.

Chen, J., A. L. Tannahill, and *M. L. Shuler.* 1985. Design of a system for the control of low dissolved oxygen concentrations: Critical oxygen concentrations for *Azotobacter vinelandii* and *Escherichia coli. Biotechnology and Bioengineering* 27:151–55.

Cho, T., G. F. Payne, and *M. L. Shuler.* 1985. Integrated product recovery and bioconversion reactors. Paper read at 7th Symposium on Biotechnology for Fuels and Chemicals, 14–17 May 1985, in Gatlinburg, TN.

Clark, D. S., J. E. Bailey, and D. D. Do. 1985. A mathematical model for restricted diffusion effects on macromolecule impregnation in porous supports. *Biotechnology and Bioengineering* 27:208.

Cohen, C., and *D.-h. Hwang.* 1985. Diffusion and relaxation in polymer-solvent systems by photon correlation spectroscopy. In *Physical optics of dynamic phenomena and processes in macromolecular systems,* ed. B. Sedláček, pp. 49–57. Berlin: Walter de Gruyter.

Eggebrecht, J., K. E. Gubbins, A. Shreve, S. M. Thompson, and *J. P. R. B. Walton.* 1985. The vapour-liquid interface for a Stockmeyer fluid. Paper read at Dense Fluids Conference, Royal Society of Chemistry, 10–12 April 1985, in Bristol, U.K.

Einsele, A., *R. K. Finn,* and W. Samhaber. 1985. *Mikrobiologische und biochemische Verfahrenstechnik: Eine Einführung.* Weinheim, Germany: Verlag Chemie.

Georgiou, G., J. J. Chalmers, M. L. Shuler, and *D. B. Wilson.* 1985. Continuous immobilized recombinant production from *E. coli* capable of selective protein excretion: A feasibility study. *Biotechnology Progress* 1:75–79.

Gray, C. G., C. G. Joslin, *V. Venkatasubramanian,* and *K. E. Gubbins.* 1985. Induction effects in fluid mixtures of dipolar-quadrupolar polarizable molecules. *Molecular Physics* 54:1129–48.

Gubbins, K. E. 1985*a*. Computer simulation and theoretical studies of liquid mixtures: Brute force vs. insight. Paper read at ChemPor '85: 4th International Chemical Engineering Conference, 15–16 April 1985, in Coimbra, Portugal.

———. 1985*b*. Molecular studies of gas properties. Paper read at Gas Research Institute Workshop on Thermophysical Properties, 21–22 March 1985, in Houston, TX.

Joslin, C. G., C. G. Gray, and *K. E. Gubbins.* 1985. Renormalized perturbation theory for dipolar and quadrupolar polarizable fluids. *Molecular Physics* 54:1117–28.

Kung, D. M., and *W. L. Olbricht.* 1985. The breakup of liquid drops in creeping flow. Paper read at 56th Annual Meeting, Society of Rheology, 25–27 February 1985, in Blacksburg, VA.

Lee, A. L., M. M. Ataai, and *M. L. Shuler.* 1985. Double substrate limited growth of *Escherichia coli. Biotechnology and Bioengineering* 26:1398–1401.

Lee, D. J., K. E. Gubbins, and *M. M. Telo da Gama.* 1985. Surface activity at the liquid-vapour interface of binary mixtures. Paper read at Dense Fluids Conference, Royal Society of Chemistry, 10–12 April 1985, in Bristol, U. K.

Lobo, L. Q., L. A. K. Staveley, *V. Venkatasubramanian, P. Clancy, K. E. Gubbins,* C. G. Gray, and C. G. Joslin. 1985. Thermodynamic properties of liquid mixtures of hydrogen chloride and tetrafluoromethane. *Fluid Phase Equilibria* 22:89–105.

Naik, C. D., P. Clancy, and *K. E. Gubbins.* 1985. The use of computer graphics to teach thermodynamic phase diagrams. *Chemical Engineering Education* 19:78–82.

Nollert, M. U., and *W. L. Olbricht.* 1985. Macromolecular deformation in periodic extensional flows. *Rheologica Acta* 24:3–14.

Nunes da Ponte, M., J. C. G. Calado, and *W. B. Streett.* 1985. Phase equilibria in Ar and CO_2 mixtures. Paper read at 9th Symposium on Thermophysical Properties, 24–27 June 1985, in Boulder, CO.

Nunes da Ponte, M., D. Chokappa, J. C. G. Calado, P. Clancy, and *W. B. Strett.* 1985. Vapor-liquid equilibrium in the xenon and ethane system. *Journal of Physical Chemistry* 89:2746–51.

Rodriguez, F. 1985*a*. Compact correlation of flow data for solutions of rigid polymer molecules. *Chemical Engineering Communications* 33:287–99.

———. 1985*b*. Positive-working electron beam resists. Paper read at IBM Lithography Symposium, 9 May 1985, in Tarrytown, NY.

———. 1985*c*. Visualizing molecular weight averages and distributions. In *Proceedings, American Chemical Society Conference on Polymeric Materials Science and Engineering,* vol. 52, pp.523–27. Washington, DC: ACS.

Rodriguez, F., C. H. Chu, W. T. W. K. Chu, and *M. A. Rondinella.* 1985. Adiabatic photopolymerization of acrylamide. *Journal of Applied Polymer Science* 30:1629–37.

Rodriguez, F., R. J. Groele, and *P. D. Krasicky.* 1985. Dissolution rates of thin polymer films using laser interferometry. In *Advances in Resist Technology and Processing.* Proceedings, Society of Photo-Optical Instrumentation Engineers, vol. 539, pp.14–20. Redondo Beach, CA: SPIE.

Rodriguez, F., P. D. Krasicky, and *R. J. Groele.* 1985. Dissolution rate measurements. *Solid State Technology* 28:125–31.

Schmidt, S. K., M. Alexander, and *M. L. Shuler.* 1985. Predicting threshold concentrations of organic substrates for bacterial growth. *Journal of Theoretical Biology* 114:1–8.

Shuler, M. L. 1985. Immobilized cell bioreactors. Paper read at New England Biotechnology Association Colloquium II, 21–22 March 1985, in Worcester, MA.

Shuler, M. L., D. B. Wilson, J. J. Chalmers, and *G. Georgiou.* 1985. Immobilized cell bioreactor for continuous recombinant protein production from *E. coli* capable of selective protein excretion. Paper read at 85th Annual Meeting, American Society for Microbiology, 3–8 March 1985, in Las Vegas, NV.

Smith, J. C. 1985. The rise of research: Fifty years of change in Cornell's engineering program. *Engineering: Cornell Quarterly* 20(1):26–36.

Smith, J. C., and *P. H. Steen.* 1985. Chemical engineering at Cornell. *Chemical Engineering Education* 19(2):58–61, 103–06.

Streett, W. B., J. A. Zollweg, and *P. Clancy.* 1985. Thermophysical property measurements to support parallel developments in computer simulation and molecular theories of fluids. Paper read at 1985 Spring Meeting, American Institute of Chemical Engineers, 24–28 March 1985, in Houston, TX.

Telo da Gama, M. M., and *K. E. Gubbins.* 1985. A microscopic model for the interfacial properties of mixtures of oil, water and non-ionic surfactants. Paper read at ChemPor '85: 4th International Chemical Engineering Conference, 15–16 April 1985, in Coimbra, Portugal.

Thurtell, J. H., M. M. Telo da Gama, and *K. E. Gubbins.* 1985. The liquid-vapour interface of simple models of nematic liquid crystals. *Molecular Physics* 54:321–32.

Tolan, J., and *R. K. Finn.* 1985. Alcohol tolerance of enterobacteria growing on pentoses. Paper read at 7th Symposium on Biotechnology for Fuels and Chemicals, 14–17 May 1985, in Gatlinburg, TN.

Zudkevitch, D., and *W. B. Streett.* 1985. Fluid mixtures at high pressures: Behavior and applications. In *The role of data in scientific progress,* ed. P. S. Glaeser, pp. 217–22. Amsterdam: Elsevier.

■ CIVIL AND ENVIRONMENTAL ENGINEERING

Barnard, T. E., and *J. J. Bisogni, Jr.* 1985. Errors in gran function analysis of titration data for dilute acidified water. *Water Research* 19(3):393–99.

Brutsaert, W. (1984.) A seminal contribution to the fluid mechanics of porous media and soils by Matano. *Nagare* [Journal of the Japan Society of Fluid Mechanics] 3(4):92–94.

———. 1985. Meteorological and hydrological experimentation in complex and hilly terrain. Invited paper read at Spring Meeting, American Geophysical Union, 27–29 May 1985, in Baltimore, MD. (Abstract in *EOS* 66:233.)

Buss, A., and *M. Grigoriu.* 1985. *Crossings of non-Gaussian processes.* Department of Structural Engineering Report no. 85–5. Ithaca, NY: Cornell University.

Charlie, W. A., J. P. Turner, and *F. H. Kulhawy.* 1985. Review of repeated axial load tests on deep foundations. In *Drilled piers and caissons,* 2 ed., ed. C. N. Baker, pp. 129–50. New York: American Society of Civil Engineers.

Dick, R. I., and *S. A. Kishbaugh.* 1985. Basic physical properties of pollution control residues: Specific resistance to filtration. In *Proceedings, Conference on Environmental Engineering and Pollution Control Processes,* ed. D. F. Carey, pp. 167–86. Washington, DC: U.S. Environmental Protection Agency.

Dworsky, L. B. 1985. Canada's water—America's needs. Paper read at Foreign Policy Conference, Canadian Institute of International Affairs, 3–5 May 1985, in Toronto, Canada.

Dworsky, L. B., and *D. J. Allee.* 1985. A national program for water resources research. Paper read at National Conference on Water Resources Research, 7–8 February 1985, in Washington, DC.

El-Kadi, A. I., and *W. Brutsaert.* 1985. Applicability of effective parameters for unsteady flow in nonuniform aquifers. *Water Resources Research* 21:183–98.

Grigoriu, M. 1985. *Response of linear systems to quadratic Gaussian excitations.* Department of Structural Engineering Report no. 85–4. Ithaca, NY: Cornell University.

Grigoriu, M., M. Khater, and *T. O'Rourke.* 1985. Stochastic beams on elastic foundations. In *Proceedings, ASCE Symposium,* ed. J. Yao, R. Corrotis, C. Brown and F. Moses, pp. 96–106. New York: American Society of Civil Engineers.

Irwin, L. H., I. Ishibashi, and *W. S. Yang.* 1985. *Validation of the mechanistic approach to nondestructive pavement evaluation for aggregate surfaced road.* U.S. Forest Service Report no. 53–56A1–00756. Milwaukee, WI: USFS.

Ishibashi, I., M. Kawamura, and S. K. Bhatia. 1985. *Effect of initial shearing on cyclic drained and undrained characteristics of sand.* National Science Foundation Report no. CEE-8314009. Washington, DC: NSF.

Kustas, W. P., and *W. Brutsaert.* 1985. The roughness characteristics of rugged hilly watersheds. Paper read at Spring Meeting, American Geophysical Union, 27–29 May 1985, in Baltimore, MD. (Abstract in *EOS* 66:261.

Liggett, J. A. 1985. Multiple boundary conditions, free surface flow, and the boundary element method. *Communications in Applied Numerical Methods* 1:105–12.

Liu, P. L.-F., S. B. Yoon, and J. T. Kirby. 1985. Nonlinear refraction-diffraction of waves in shallow water. *Journal of Fluid Mechanics* 153:185–201.

Orloff, N. 1985. Commentary: Why EPA's approach to toxic chemicals doesn't work. *Engineering: Cornell Quarterly* 20(1):37–39.

O'Rourke, T. D., and *F. H. Kulhawy.* 1985. Observations on load tests for drilled shafts. In

Drilled piers and caissons, 2 ed., pp. 113–28. New York: American Society of Civil Engineers.

Perdiharis, P. C., S. Hilmy, and *R. N. White*. 1985. Extensional stiffness of precracked reinforced concrete. *ASCE Journal of Structural Engineering* 111(3):487–504.

Perdiharis, P. C., and *R. N. White*. 1985. Shear modulus of precracked reinforced concrete. *ASCE Journal of Structural Engineering* 111(2):270–89.

Philipson, W. R., D. R. Gordon, W. D. Philpot, and *V. L. Williams*. 1985. Calibration for radiometric measurements with non-white reflectance standards. In *Proceedings, 51st Annual Meeting, American Society of Photogrammetry*, pp. 47–54. Falls Church, VA: ASP.

Philipson, W. R., V. L. Williams, D. K. Gordon, and *W. Philpot*. 1985. Vegetable and fruit tree inventory with Landsat TM data. In *Proceedings, 51st Annual Meeting, American Society of Photogrammetry*, pp. 39–46. Falls Church, VA: ASP.

Philpot, W. 1985. *Experimental verification of a radiative transfer model for assessment of water quality*. NSF report no. PB-85-181519. Springfield, VA: National Technical Information Service.

Philpot, W. and *W. R. Philipson*. 1985. Thermal sensing for characterizing the contents of waste storage drums. *Photogrammetric Engineering and Remote Sensing* 51:237–43.

Stedinger, J. R., D. P. Lettenmaier, and *R. M. Vogel*. 1985. Multisite ARMA(1,1) and disaggregation models for annual streamflow generation. *Water Resources Research* 21(4):497–510.

Stedinger, J. R., D. Rei, and *T. A. Cohn*. 1985. A condensed disaggregation model for incorporating parameter uncertainty into monthly reservoir simulations. *Water Resources Research* 21(5):665–75.

Steifel, R. C., *L. B. Dworsky, et al.* 1985. *Research and development: Water resources research in the FY 1986 budget*. American Association for the Advancement of Science report no. 10. Washington, DC: AAAS.

Taigbenu, A. E., and *J. A. Liggett*. 1985. Boundary element calculations of the diffusion equation. *ASCE Journal of the Engineering Mechanics Division* 111:311–28.

Turnquist, M. A., and W. C. Jordan. 1985. *Fleet sizing under production cycles and uncertain travel times*. General Motors Research Labs report no. GMR-5012.

Vodacek, A., and *W. Philpot*. 1985. Use of induced fluorescence measurements to assess aluminum-organic interactions in acidified lakes. In *Proceedings, 51st Annual Meeting, American Society of Photogrammetry*, pp. 460–69. Falls Church, VA: ASP.

Vogel, R. M., and *J. R. Stedinger*. 1985. Minimum variance streamflow record augmentation procedures. *Water Resources Research* 21(5):715–23.

Willmott, C. J., C. M. Rowe, and *W. Philpot*. 1985. Small-scale climate maps: A sensitivity analysis of some common assumptions associated with grid point interpolation and contouring. *The American Cartographer* 12:5–16.

Wu, C.-S., and *P. L.-F. Liu*. 1985. Finite element modeling of nonlinear coastal currents. *ASCE Journal of Waterway, Port, Coastal, and Ocean Engineering* 111(2):417–32.

■ COMPUTER SCIENCE

Alford, M. W., J. P. Ansart, G. Hommel, L. Lampert, B. Liskov, G. P. Mullery, and *F. B. Schneider*. 1985. *Distributed systems: Methods and tools for specification*. Heidelberg: Springer-Verlag.

Bilardi, G., and F. P. Preparata. 1985*a*. A minimum area VLSI network for 0(log N) time sorting. *IEEE Transactions on Computers* C-34(4):336–43.

———. 1985*b*. Tessellation techniques for area-time lower bounds. In *Proceedings, 19th Annual Conference on Information Sciences and Systems*, pp. 7–9. Baltimore, MD: The Johns Hopkins University Press.

———. 1985*c*. The VLSI optimality of the AKS sorting network. *Information Processing Letters* 20(2):55–59.

Gries, D. and *J. Prins*. 1985. A new notion of encapsulation. Paper read at SIGPLAN 85 Symposium on Language Issues in Programming Environments, 23–25 June 1985, in Seattle, WA.

Hopcroft, J., and A. Borodin. 1985. Routing, merging and sorting on parallel models of computation. *Journal of Computer and System Sciences* 30(1):130–45.

Hoffmann, C., and *J. Hopcroft*. 1985. Quadratic blending surfaces. Paper read at Society for Industrial and Applied Mathematics-Rensselaer Polytechnic Institute Conference, 15–18 July 1985, in Albany, NY.

Jacobs, D., and *D. Gries*. 1985. General correctness: A unification of partial and total correctness. *Acta Informatica* 22:67–84.

Johnson, G. F., and C. N. Fischer. 1985. A metalanguage and system for nonlocal attribute flow in language-based editors. In *Proceedings, 12th ACM Symposium on Principles of Programming Languages*, ed. B. K. Reid, pp. 141–51. New York: Association for Computing Machinery.

Nguyen, V., A. Demers, D. Gries, and S. Owicki. 1985. Behavior: A temporal approach to process modeling. Paper read at Conference on Logics of Programs, 12–13 June 1985, in Brooklyn, NY.

Nguyen, V., D. Gries, and S. Owicki. 1985. A model and temporal proof system for networks of processes. Paper read at Annual Symposium on Principles of Programming Languages, 23–25 January 1985, in New Orleans, LA.

Salton, G. 1985. A note on information retrieval models and theories. In *Recherche d'Information Assistée par Ordinateurs*, pp. 1–27. Proceedings, RIAO 85 Conference. Grenoble, France: RIAO.

Salton, G., E. A. Fox, and *E. Voorhees*. 1985. Advanced feedback methods in information retrieval. *Journal of the American Society for Information Science* 36(3):200–10.

Teitelbaum, T., and *S. Horwitz*. 1985. Relations and attributes: A symbiotic basis for editing environments. In *SIGPLAN 85 Symposium on Language Issues in Programming Environments*. ACM Special Interest Group on Programming Languages, vol. 20, no. 7, pp. 93–106. New York: Association for Computing Machinery.

Valiant, L. G., and *V. V. Vazirani*. 1985. NP is as easy as detecting unique solutions. In *Proceedings, 17th Annual Symposium on Theory of Computing*, pp. 458–63. New York: Association for Computing Machinery.

Vazirani, U. V., and *V. V. Vazirani*. 1985. The two-processor scheduling problem is in R-NC. In *Proceedings, 17th Annual Symposium on Theory of Computing*, pp. 11–21. New York: Association for Computing Machinery.

■ ELECTRICAL ENGINEERING

Adesida, I., E. Kratschmer, E. D. Wolf, A. Muray, and *M. Isaacson*. 1985. Ion beam lithography at nanometer dimensions. *Journal of Vacuum Science and Technology* B3(1):45–49.

Berger, T., and *J.-C. Huang*. 1985. Delay analysis of interval-searching contention resolution algorithms. *IEEE Transactions on Information Theory* IT-31:264–74.

Brown, A. S., S. C. Palmateer, G. W. Wicks, L. F. Eastman, and *A. R. Calawa*. 1985. The behavior of unintentional impurities in $Ga_{.47}In_{.53}As$ grown by MBE. *Journal of Electronic Materials* 14(3):367–78.

Byrne, D. M., A. J. Brouns, F. C. Case, *R. C. Tiberio, B. L. Whitehead,* and *E. D. Wolf*. 1985. Infrared mask filters fabricated by electron beam lithography. *Journal of Vacuum Science and Technology* B3(1):268–71.

Capani, P. M., S. D. Mukherjee, L. Rathbun, H. T. Griem, G. W. Wicks, and *L. F. Eastman*. 1985. The characterization of alloyed NiGeAuAgAu ohmic contacts to the GaInAs/AlInAs heterostructure by Auger electron spectroscopy sputter depth profiling. Paper read at Materials Research Society Conference, 15–18 April 1985, in San Francisco, CA.

Chan, K. T., L. D. Zhu, and *J. M. Ballantyne*.

1985. Growth of high quality GaInAs on InP buffer layers by MOCVD. *Applied Physics Letters* 47:44–48.

Chinn, J. D., and *E. D. Wolf.* (1984.) Characteristics of reactive ion beam and ion assisted etching using direct ion mass analysis and emission spectroscopy. Paper read at 166th Electrochemical Society Meeting, 16 October 1984, in New Orleans, LA.

———. 1985. The role of reactive ions in ion-beam assisted etching. *Journal of Vacuum Science and Technology* B3(1):410–15.

Delchamps, D. F. 1985. New approaches to the sensitivity analysis of feedback systems. In *Proceedings, 19th Annual Conference on Information Science and Systems,* pp.476–82. Baltimore, MD: The Johns Hopkins University Press.

Eastman, L. F. 1985*a*. Compound semiconductor high speed electron devices. Paper read at Cornell Electrical Engineering Centennial, 15 March 1985, in Stanford, CA.

———. 1985*b*. Compound semiconductor materials growth and properties and compound semiconductor microwave transistors. Invited paper read at International Workshop on Digital and Analog GaAs MMIC Devices and Applications, 28–31 May 1985, in Rome, Italy.

———. 1985*c*. Device physics. Paper read at conference, American Physical Society, 25–29 March 1985, in Baltimore, MD.

———. 1985*d*. Recent results in microelectronics research at Cornell: A layman's view. Paper read at alumni talk, 22 April 1985, in Rhinebeck, NY.

Erskine, D. J., C. L. Tang, and *A. J. Taylor.* (1984.) Dynamic Burstein-Moss shift in GaAs and GaAs/AlGaAs multiple quantum well structures. *Applied Physics Letters* 45:1209–11.

Frey, J. 1985. Materials engineering for semiconductor devices. Paper read at Annual Meeting, Electrochemical Society of Japan, 5 April 1985, in Kofu, Japan.

Frey, J., and K. Bhasin. 1985. Microwave integrated circuits, 2nd ed. Boston: Artech House.

Gharachorloo, N., and *C. Pottle.* 1985. Super buffer: A systolic VLSI graphics engine for real time raster image generation. In *1985 Chapel Hill Conference on Very Large Scale Integration,* ed. H. Fuchs, pp. 285–305. Rockville, MD: Computer Science Press.

Gosnell, T. R., A. J. Sievers, and *C. R. Pollock.* 1985. Continuous wave operation of the KBr:CN- solid state vibration laser in the 5μm region. *Optics Letters* 10:125–27.

Guillen, M. A., and *R. L. Liboff.* (1984.) Unified kinetic theory of plasma correlations. *Journal of Plasma Physics* 32(1):81–98.

Hagfors, T. 1985. The Arecibo observatory: Past achievements and future prospects in the exploration of the ionosphere and the solar system. Paper read at Electrical Engineering Centennial Celebration, 17–18 April 1985, in St. Louis, MO.

Itoh, T., T. Griem, G. W. Wicks, and *L. F. Eastman.* 1985. Sheet electron concentration at the hetero-interface $Al_{0.48}In_{0.52}As/Ga_{0.47}In_{0.53}As$ modulation doped structures. *Electronics Letters* 21(9):373–74.

Itoh, T., G. W. Wicks, and *L. F. Eastman.* 1985. Progress toward InSb FET's on semi-insulating substrates. Paper read at Workshop on Compound Semiconductors for Microwave Materials and Devices, 11–13 February 1985, in Fort Lauderdale, FL.

Krusius, J. P. 1985*a*. Process integration for 1/4 micrometer MOS technologies. Invited paper read at IEEE Electron Devices Section Meeting, 5 March 1985, in Rochester, NY.

———. 1985*b*. Toward gigabit logic with silicon. Invited paper read at General Motors Research Laboratories, 26 February 1985, in Warren, MI.

Krusius, J. P., J. Nulman, and *A. Perera.* 1985. Self-aligned dual surface lithography. Paper read at 29th International Symposium on Electron, Ion, and Photon Beams, 28–31 May 1985, in Portland, OR.

Lee, T.-A., and *C. Heegard.* 1985. An inversion technique for the design of binary convolutional codes for the I-D^N channel. Paper read at 1985 Conference on Information Sciences and Systems, 27–29 March 1985, in Baltimore, MD.

Li, J. Z., I. Adesida, and *E. D. Wolf.* 1985. Profile control for submicron structures in GaAs by reactive ion etching using $SiCl_4$. *Journal of Vacuum Science and Technology* B3(1):406–09.

Liboff, R. L. (1984*a*.) An elementary deviation of a group-theoretic property of many-particle spin states. *American Journal of Physics* 52:561.

———. (1984*b*.) The correspondence principle revisted. *Physics Today* 37:50–55.

———. (1984*c*.) Criteria for physical domains in laboratory and solid-state plasmas. *Journal of Applied Physics* 56:3530–35.

———. 1985*a*. Generalized Bogoliukov hypothesis for dense fluids. *Physical Review* 31:1883–89.

———. 1985*b*. Geometrical properties of the Fermi energy. *Foundations of Physics* 15:339–52.

———. 1985*c*. Plasma domains in extrinsic GaAs and InP. *Journal of Physics and Chemistry of Solids* 46:103–05.

Luk, F. T. 1985*a*. On the minres method of factor analysis. *SIAM Journal on Scientific and Statistical Computing* 6:562–72.

———. 1985*b*. A parallel method for computing the generalized singular value decomposition. In *Proceedings, 7th Symposium on Computer Arithmetic,* ed. K. Hwang, pp. 260–65. NY: Institute of Electrical and Electronics Engineers.

Lunardi, L. M., P. M. Enquist, P. J. Tasker, and *L. F. Eastman.* 1985. Microwave characteristics of an (AlGa)As/GaAs heterojunction bopolar transistor. Paper read at Workshop on Compound Semiconductors for Microwave Materials and Devices, 11–13 Februrary 1985, in Fort Lauderdale, FL.

McCombe, B. D., B. V. Shanabrook, J. Comas, *J. Ralston,* and *G. Wicks.* 1985. Binding of shallow donor impurities in quantum-well structures. *Physical Review Letters* 54(12):1283–86.

Mukherjee, S. D., and *D. W. Woodard.* 1985. *Etching and surface preparation of GaAs for device fabrication.* New York: Wiley.

Nastasi, M., R. Fastow, J. Gyulai, *J. W. Mayer, S. J. Plimpton,* and *E. D. Wolf.* (1984.) Ion induced reactions in Fe/As bilayers by pulsed beam ion irradiation and Xe implantation. Paper read at 1984 Conference on Ion Beam Modification of Materials, 16–20 July 1984, at Cornell University, Ithaca, NY.

Nulman, J., J. P. Krusius, and A. Gat. 1985. Rapid thermal processing of thin gate dielectrics: Oxidation of silicon. *IEEE Electron Device Letters* EDL-6:205–07.

Pollock, C. R., J. F. Pinto, and *L. W. Stratton.* 1985. Stable color center laser in K-doped NaCl tunable from 1.41 to 1.76 μm. Paper read at Optical Society of America Conference on Tunable Solid State Lasers, 15–16 May 1985, in Arlington, VA.

Rosker, M., and *C. L. Tang.* 1985. Widely tunable optical parametric oscillator using urea. *Journal of the Optical Society of America* B2:691–96.

Sethares, W. A., D. A. Lawrence, C. R. Johnson, Jr., and *R. R. Bitmead.* 1985. On the existence of unbounded parameters in adaptive identificaiton. In *Proceedings, 19th Conference on Information Sciences and Systems,* pp. 38–43. Baltimore, MD: The Johns Hopkins University Press.

Sonek, G. J., and *J. M. Ballantyne.* (1984.) Reactive ion etching of GaAs using BCl_3. *Journal of Vacuum Science and Technology* B2(4):653.

Tang, C. L. (1984.) Femtosecond solid state measurements. Invited paper read at Annual Meeting, Optical Society of America, November 1984, in San Diego, CA.

Tang, C. L., M. Rosker, and K. Cheng. 1985. A broadly tunable urea optical parametric oscillator. Invited paper read at Conference on Lasers and Electro-Optics, 21–24 May 1985, in Baltimore, MD.

Taylor, A. J., D. J. Erskine, and *C. L. Tang.* 1985. Ultrafast relaxation dynamics of photo-excited carriers in GaAs and related compounds. *Journal of the Optical Society of America* B2:663–73.

von Lehmen, A., and *J. M. Ballantyne.* (1984.) Picosecond luminescence measurements on fast GaAs Schottky diodes under changing circuit conditions. *Applied Physics Letters* 45:767–69.

———. 1985. Investigation of the nonlinearity in the luminescence of GaAs under high density picosecond photoexcitation. *Journal of Applied Physics* 58:958.

Welch, D. F., G. W. Wicks, and *L. F. Eastman.* 1985. Luminescence lineshape broadening mechanisms in GaInAs/AlInAs quantum wells. *Applied Physics Letters* 46(10):991–93.

Wolf, E. D. (1984*a*.) Microfabrication research at the National Submicron Facility. Invited papers read at 31st National Symposium of the American Vacuum Society, 4–7 December 1984, in Reno, NV.

———. (1984*b*.) Microminiaturization: Fanning the microelectronics revolution. 10th Annual Mohler Lecture, 23 October 1984, at McPherson College, McPherson, KS.

———. 1985*a*. Chemistry and physics of microstructures fabrication. Invited lecture given at Chemistry Colloquium Series, 21 February 1985, at University of North Carolina, Chapel Hill, NC.

———. 1985*b*. Research at the National Submicron Facility. Invited lecture, Distinguished Speaker Series, Microelectronics Center of North Carolina, 22 February 1985, in Raleigh, NC.

———. 1985*c*. Submicron electronics. Invited paper read at 1984–85 Grumman–University Technology Forum, 11–12 February 1985, in Bethpage, NY.

Wong, S., and W. Oldham. 1985. Anodic nitridation of silicon and silicon dioxide. *IEEE Transactions on Electron Devices* ED-32:978.

Zhu, L. D., K. T. Chan, D. K. Wagner, and *J. M. Ballantyne.* 1985. Photoluminescence study of the growth of indium phosphide by MOCVD. *Journal of Applied Physics* 57:5486.

Zhu, L. D., P. Sulewski, K. T. Chan, and *J. M. Ballantyne.* 1985. 2DEG in $In_{0.53}Ga_{0.47}As/InP$ heterostructures grown by atmospheric pressure MOCVD. Paper read at March Meeting, American Physical Society, 25–29 March 1985, in Baltimore, MD.

■ GEOLOGICAL SCIENCES

Allegre, C. J., and *D. L. Turcotte.* 1985. Geodynamic mixing in the mesosphere boundary layer and the origin of oceanic islands. *Geophysics Research Letters* 12:207–10.

Allmendinger, R. W., and *T. E. Jordan.* (1984.) Mesozoic structure of the Newfoundland Mountains, Utah: Horizontal shortening and subsequent extension in the hinterland of the Sevier belt. *Bulletin of the Geological Society of America* 95:1280–92.

Arnow, J., K. D. Nelson, J. McBride, J. Oliver, L. Brown, and *S. Kaufman.* 1985. Location and character of the late Paleozoic suture beneath the southeastern U.S. coastal plain: Evidence from new COCORP profiling. Paper read at meeting of American Geophysical Union, 27 May–1 June 1985, in Baltimore, MD. Abstract in *EOS* 66:358.

Bassett, W. A. 1985. High pressure-temperature x-ray diffraction using synchrotron radiation. *Nuclear Instruments and Methods in Physics Research* B10/11:309–12.

Bassett, W. A., M. D. Furnish, and *E. Huang.* 1985. Applications of synchrotron radiation in high pressure-temperature mineralogy. In *Solid state physics under pressure,* ed. S. Minomura, pp.335–41. Tokyo: Terra Scientific.

Brown, L. D. 1985. Vertical crustal motion and the North American vertical datum. In *Proceedings, 3rd International Symposium on the North American Vertical Datum,* ed. D. B. Zilkoski, pp. 207–16. Rockville, MD: National Geodetic Survey.

Cimino, J. B., *A. Bloom,* J. Rabassa, and S. Wall. 1985. Multiple incidence angle SIR-B imagery of the Chubut Province of southern Argentina. Paper read at International Symposium on Remote Sensing of Environment, 1–4 April 1985, in San Francisco, CA.

Kay, R. W. 1985. Island arc processes relevant to crustal and mantle evolution. *Tectonophysics* 112:1–15.

Kay, R. W., and *S. M. Kay.* 1985. Eclogite model and primary magmas of the Aleutian arc. Paper read at meeting of American Geopysical Union, 27 May–1 June 1985, in Baltimore, MD. Abstract in *EOS* 66:422.

Kay, S., V. Maksaev, and C. Gordillo. 1985. Middle-late Tertiary volcanism over a section of the modern shallowly-dipping Andean Benioff zone (29–32°S). Paper read at American Geophysical Union Meeting, 27 May–1 June 1985, in Baltimore, MD. Abstract in *EOS* 66:422.

Klemperer, S., and *L. Brown.* 1985. Simulations of noise rejection and mantissa-only recording: An experiment in high-amplitude noise reduction with COCORP data. *Geophysics* 50:709–14.

Klemperer, S. L., L. Brown, J. Oliver, C. Ando, B. Czuchra, and *S. Kaufman.* 1985. Some results of COCORP seismic reflection profiling in the Grenville-age Adirondack Mountains, New York State. *Canadian Journal of Earth Sciences* 22:141–53.

McBride, J., K. D. Nelson, J. Arnow, J. Oliver, L. Brown, and *S. Kaufman.* 1985. New COCORP profiling in the southeastern U.S. coastal plain: Late Paleozoic suture and Mesozoic rift basin. Paper read at meeting of American Geopysical Union, 27 May–1 June 1985, in Baltimore, MD. Abstract in *EOS* 66:358.

Nelson, K. D. 1985. Suture and rift basin evident from new COCORP profiles on the southeastern U.S. coastal plain. Paper read at Geological Society of America, Penrose Conference, 27 May–3 June 1985, in Liscombe Mills, Nova Scotia.

Nelson, K. D., T. F. Zhu, A. Gibbs, R. Harris, J. Oliver, S. Kaufman, and *L. Brown.* (1984.) COCORP deep seismic reflection profiling in the northern Sierra Nevada Mountains, California. Paper read at annual meeting, Geological Society of America, 5–8 November 1984, in Reno, NV.

Ni, J., and *M. Barazangi.* 1985. Active tectonics of the western Tethyan Himalaya above the underthrusting Indian plate: The upper Sutlej river basin as a pull-apart structure. *Tectonophysics* 112:277–95.

Ockendon, J. R., A. B. Taylor, *S. H. Emerman,* and *D. Turcotte.* 1985. Geodynamic thermal runaway with melting. *Journal of Fluid Mechanics* 152:301–14.

Schweller, W. J., P. H. Roth, *D. E. Karig,* and *S. B. Bachman.* (1984.) Sedimentation history and biostratigraphy of ophiolite-related Tertiary sediments, Luzon, Philippines. *Geological Society of America Bulletin* 95:1333–42.

Taylor, F. W., C. Jouannic, and *A. L. Bloom.* 1985. Quaternary uplift of the Torres Islands, northern New Hebrides frontal arc: Comparison with Santo and Malekula Islands, central New Hebrides frontal arc. *Journal of Geology* 93:419–38.

Turcotte, D. L. 1985. Fractals, fragmentation, and a renormalization group determination of the frequency-mass distribution of meteorites and asteroids. Paper read at 16th Lunar and Planetary Science Conference, 11–15 March 1985, in Houston, TX.

Turcotte, D. L., and J. C. Pflugrath. 1985. Thermal structure of the accreting earth. *Journal of Geophysical Research* 90:C541–C544.

■ MATERIALS SCIENCE AND ENGINEERING

Ast, D. G. (1984.) a-Si: H FET-addressed LCD panel. In *Semiconductors and semimetals,* vol. 21, pt. d, pp. 115–38. New York: Academic Press.

Baik, S., D. E. Fowler, J. M. Blakely, and *R. Raj.* 1985. Segregation of Mg to the (0001) surface of doped sapphire. *Journal of the American Ceramic Society* 68:281–86.

Baik, S., and *R. Raj.* 1985. Effect of silicon activity on liquid phase sintering of nitrogen ceramics. *Journal of the American Ceramic Society* 78:124–26.

Barbour, J. C., F. W. Saris, M. Nastasi, and *J. W. Mayer.* 1985. Amorphous Ni-Zr alloys as barriers for self-diffusion. *Physical Review B* 32:1363–65.

Birecki, H., S. Naberhuis, T. Anthony, and *D. G. Ast.* 1985. Magnetooptic quadrilayer reliability and performance. In *Optical mass data storage,* p. 19. Proceedings, Society of Photo-Optical Instrumentation Engineers, vol. 529. Bellingham, WA: SPIE.

Bordia, R. K., and *R. Raj.* 1985. Sintering behavior of ceramic films constrained by a rigid substrate. *Journal of the American Ceramic Society* 68:287–92.

Brister, K. E., Y. H. Vohra, and *A. L. Ruoff.* 1985. High-pressure phase transition in CsCl at $V/V_o = .53$. *Physical Review B* 31:4657–58.

Carter, C. B. (1984.) What's new in dislocation dissociation. In *Comptes rendus du colloque international du CNRS dislocations: Structure de coeur et propriétés physiques.* Paris, France: Editions du CNRS.

Chen, S. H., J. C. Barbour, L. R. Zheng, C. B. Carter, and *J. W. Mayer.* 1985. Structure analysis of Ni-silicides formed in lateral diffusion couples. In *Proceedings, Materials Research Society Symposia,* vol. 37, pp. 635–40. Pittsburgh, PA: MRS.

Chiang, S.-W., and *D. L. Kohlstedt.* 1985. Load relaxation studies in germanium. *Journal of Materials Science* 20:736–55.

Christensen, T. M. 1985*a. Annotated bibliography of ellipsometry and optical properties of solids.* Materials Science Center Report no. 5615. Ithaca, NY: Cornell Unversity.

———. 1985*b. Annotated bibliography of metal oxidation.* Materials Science Center Report no. 5616. Ithaca, NY: Cornell University.

Cooper, R. F., and *D. L. Kohlstedt.* (1984*a.*) Sintering of olivine and olivine-basalt aggregates. *Physics and Chemistry of Minerals* 11:5–16.

———. (1984*b.*) Solution-precipitation creep in olivine-basalt aggregates. *Tectonophysics* 107:207–33.

DeMott, G. J., and *D. L. Kohlstedt.* 1985*a.* Sol-gel synthesis of fayalite. Paper read at spring meeting, American Chemical Society, 6–10 May 1985, in Cincinnati, OH.

———. 1985*b.* Surface modification of alumina by sol-gel technique. Paper read at spring meeting, American Chemical Society, 6–10 May 1985, in Cincinnati, OH.

Dimos, D., D. L. Kohlstedt, and H. Schmalzried. 1985. Kinetic demixing in a stress gradient. Paper read at spring meeting, American Chemical Society, 6–10 May 1985, in Cincinnati, OH.

Durham, W. B., and *D. L. Kohlstedt.* (1984.) Observations of shape change parameters in single crystal olivine with application to dislocation climb. Paper read at fall meeting, American Geophysical Union, 2–8 December 1984, in San Francisco, CA.

Galvin, G. L., J. W. Mayer, and P. S. Peercy. 1985. Solidification kinetics of pulsed laser melted silicon based on thermodynamic considerations. *Applied Physics Letters* 46:644–46.

Gleichmann, R., B. Cunningham, and *D. G. Ast.* 1985. Process-induced defects in solar cell silicon. *Journal of Applied Physics* 58:223–29.

Gleichmann, R., J. P. Kalejs, and *D. G. Ast.* 1985. EBIC evidence for carbon-based gettering in EFG silicon. In *Proceedings, Materials Research Society Symposia,* vol. 36, pp. 181–86. Pittsburgh, PA: Materials Research Society.

Gleichmann, R., M. D. Vaudin, and *D. G. Ast.* 1985. Recovery of edge-defined film-fed grown silicon: Dislocation/twin boundary interaction and mechanisms for twin-induced grain boundary formation. *Philosophical Magazine A* 51:449–67.

Green, P. F., P. J. Mills, C. J. Palmstrom, J. W. Mayer, and *E. J. Kramer.* 1985. Ion beam analysis of diffusion in polymer melts. In *Electronic packaging materials science,* ed. E. A. Giess, K.-N. Tu, and D. R. Uhlmann, pp. 265–70. Pittsburg, PA: Materials Research Society.

Green, P. F., C. J. Palmstrom, J. W. Mayer, and *E. J. Kramer.* 1985. Marker displacement measurements of polymer-polymer interdiffusion. *Macromolecules* 18:501–07.

Grubb, D. T. (1984*a.*) On shrinkage and molecular extension. *Journal of Materials Science Letters* 3:499–502.

———. (1984*b.*) Radiation damage of organic materials in the transmission electron microscope. *Ultramicroscopy* 12:279–80.

Grubb, D. T., and *J. J.-H. Liu.* (1984.) Real time small-angle x-ray scattering during annealing of polymer single crystals. *Journal of Polymer Science* 22:367–78.

Hannula, S.-P., D. Stone, and *C.-Y. Li.* 1985. The use of indentation techniques in evaluating mechanical properties of thin film metallizations. In *Proceedings, 35th Electronic Components Conference,* p. 424. New York: Institute of Electrical and Electronics Engineers.

Herschitz, R., and *D. N. Seidman.* 1985. Radiation-induced precipitation in fast-neutron irradiated tungsten-rhenium alloys: An atom-probe field-ion microscope study. *Nuclear Instruments and Methods in Physics Research* B718:137–42.

Huang, T.-L., and *A. L. Ruoff.* 1985. High-pressure-induced phase transitions of mercury chalcogenides. *Physical Review B* 31:5976–83.

Hung, L. S., and *J. W. Mayer.* 1985. Marker experiments in growth studies of Ni_2Si, Pd_2Si, and $CrSi_2$ formed by thermal annealing and by ion mixing. *Journal of Applied Physics* 58:1527–36.

Kavanagh, K., S. H. Chen, C. Palmstrom, C. B. Carter, and *S. D. Mukherjee.* (1984.) RBS and TEM analysis of Ta silicides of GaAs. In *Proceedings, Materials Research Society Symposia,* vol. 25, pp. 143–48. Pittsburgh, PA: MRS.

Kohlstedt, D. L. 1985*a.* Electron diffraction and weak-beam electron micrscopy. Invited paper read at Mineralogical Association of Canada 1985 Short Course on Applications of Electron Microscopy in the Earth Sciences, 12–14 May 1985, in Fredericton, New Brunswick, Canada.

———. 1985*b.* Structure and rheology of partially molten rocks. Invited seminar given at Conference on Melt and the Mantle, 29–30 April 1985, at Woods Hole Oceanographic Institute, Woods Hole, MA.

Kohlstedt, D. L., and *R. F. Cooper.* (1984.) Rheology and structure of olivine-basalt partial melts. Invited seminar given at U.S.–Japan Joint Seminar on Partial Melting, 4–8 September 1984, in Eugene, OR.

———. 1985. Flow of olivine-basalt partial melts. Invited paper read at spring meeting, American Geophysical Union, 27–31 May 1985, in Baltimore, MD.

Kohlstedt, D. L., and *D. L. Ricoult.* (1984.) High-temperature creep of silicate olivines. In *Deformation of ceramic materials: II,* ed. R. Radt and R. Tressler, pp. 251–80. New York: Plenum.

Kramer, E. J. 1985. Polymer-polymer interdiffusion. In *Electronic packaging materials science,* ed. E. A. Giess, K.-N. Tu, and D. R. Uhlmann, pp. 227–37. Pittsburgh, PA: Materials Research Society.

Kuester, K.-H., B. C. De Cooman, and *C. B. Carter.* 1985. Dislocation motion in GaAs and AlGaAs/GaAs devices. In *Proceedings, 13th International Conference on Defects in Semiconductors,* ed. L. C. Kimerling and J. M. Parsey, pp. 351–57. New York: American Institute of Mining, Metallurgical and Petroleum Engineers.

Kuo, C. C., S. L. Phoenix, and *E. J. Kramer.* 1985. Geometrically necessary entanglement loss during crazing of polymers. *Journal of Materials Science Letters* 4:459–62.

List, F. A., and *J. M. Blakely.* 1985. Kinetics of CO formation on singular and stepped Ni surfaces. *Surface Science* 152/153:463–70.

Mackwell, S. J., D. L. Kohlstedt, and M. S. Paterson. 1985. Water weakening of olivine single crystals. Paper read at spring meeting, American Geophysical Union, 27–31 May 1985, in Baltimore, MD.

Milkove, K. R. 1985. Theoretical consideration of the Volmer-Weber growth mechanism for thin epitaxial films. In *Proceedings, Materials Research Society Symposia,* vol. 37, pp. 89–98. Pittsburgh, PA: MRS.

Milkove, K. R., P. A. Lamarre, F. Schmückle, M. D. Vaudin, and *S. L. Sass.* 1985. Diffraction studies of the atomic structure of grain boundar-

ies. In *Proceedings, Materials Research Society Symposia,* vol. 41, 195–206. Pittsburgh, PA: MRS.

Milkove, K. R., and *S. L. Sass.* 1985. Experimental observations on the growth of epitaxial films by the Volmer-Weber mechanism. In *Proceedings, Materials Research Society Symposia,* vol. 37, pp. 83–88. Pittsburgh, PA: MRS.

Morel, D. E., and *D. T. Grubb.* (1984*a*.) Craze behavior in isotactic polystyrene: 1. Craze-spherulite interaction. *Polymer* 25:417–29.

———. (1984*b*.) Staining of melt crystallized isotactic polystyrene by RuO_4. *Polymer Communications* 25:68–71.

Morrisey, K. J., and *C. B. Carter.* 1985. Characterization of grain boundaries in alumina. In *Advances in materials characterization: II,* ed. R. L Snyder, R. A. Condrate, and P. F. Johnson, pp. 179–87. New York: Plenum.

Morrissey, K. J., K. K. Czanderna, R. P. Merrill, and *C. B. Carter.* (1984.) Analysis of transition alumina structures using HRTEM. In *Proceedings, 42nd Annual Meeting, Electron Microscopy Society of America,* pp. 648–49. McLean, VA: EMSA.

Nastasi, M., L. S. Hung, H. H. Johnson, and *J. W. Mayer.* 1985. Phase transformation of Ni_2Al_3 to NiAl: I. Ion irradiation induced. *Journal of Applied Physics* 57:1050–54.

Palmstrom, C. J., and D. V. Morgan. 1985. Metallization for GaAs devices and circuits. In *Gallium arsenide,* ed. M. J. Howes and D. V. Morgan, pp. 195–261. New York: Wiley.

Quinn, C. J., and *D. L. Kohlstedt.* (1984*a*.) Reactive processing of titanium carbide with titanium: I. Liquid phase sintering. *Journal of Materials Science* 19:1229–41.

———. (1984*b*.) Reactive processing of titanium carbide with titanium: II. Solid-state hot pressing. *Journal of Materials Science* 19:1242–50.

———. (1984*c*.) The solid-state reaction between titanium carbide and titanium. *Journal of the American Ceramic Society* 67:305–10.

Raj, R., and F. F. Lange. 1985. On the retention of equiaxed grain structure after superplastic and other forms of high temperature deformation. *Acta Metallurgica* 33:699–703.

Ricoult, D. L., and *D. L. Kohlstedt.* (1984.) Effect of chemical environment on high-temperature creep of olivine. Paper read at 10th Annual Meeting, European Geophysical Society, 30 July–3 August 1984, in Louvain-la-Neuve, Belgium.

Saris, F. W., L. S. Hung, M. Nastasi, J. W. Mayer, and *B. Whitehead.* 1985. Failure temperature of amorphous Cu-Ta alloys as diffusion barriers in Al-Si contacts. *Applied Physics Letters* 46:646–48.

Schaff, W. J., *P. A. Maki, L. F. Eastman, L. Rathbun, B. C. De Cooman,* and *C. B. Carter.* 1985. The effect of doping on the interface between GaAs and AlGaAs. In *Proceedings, Materials Research Society Symposia,* vol. 37, pp. 15–21. Pittsburgh, PA: MRS.

Sickafus, K., and *S. L. Sass.* 1985. Grain boundary structural transformations induced by solute segregation. In *Proceedings, Materials Research Society Symposia,* vol. 41, pp. 207–12. Pittsburgh, PA: MRS.

Skrotski, W., H. Wendt, C. B. Carter, and *D. L. Kohlstedt.* (1984*a*.) Structural changes of a $\Sigma = 51$ tilt boundary in Ge during high-temperature creep. Paper read at fall meeting, Materials Research Society, November 1984, in Boston, MA.

———. (1984*b*.) Structure and dissociation of a 15° <110> tilt boundary in Ge. In *Proceedings, Materials Research Society Symposium,* vol. 25, pp. 299–304. Pittsburgh, PA: MRS.

Stone, D., S.-P. Hannula, and *C.-Y. Li.* 1985. The effects of service and material variables on the fatigue behavior of solder joints during the thermal cycle. In *Proceedings, 35th Electronic Components Conference,* p. 46. New York: Institute of Electrical and Electronics Engineers.

Vaudin, M. D., F. Schmückle, P. A. Lamarre, and *S. L. Sass.* 1985. Analysis of the structure of grain boundaries normal to the boundary plane using diffraction techniques. In *Proceedings, Materials Research Society Symposia,* vol. 41, pp. 221–26. Pittsburgh, PA: MRS.

Vohra, Y. K., S. J. Duclos, and *A. L. Ruoff.* 1985*a*. Instability of the CsCl structure in ionic solids at high pressures. *Physical Review Letters* 54:570–73.

———. 1985*b*. The Madelung constant for the tetragonal distortion of the CsCl lattice. *Journal of Physics and Chemistry of Solids* 46:515–17.

Whetten, T. J., and *A. L. Ruoff.* 1985. Segregation of copper in aluminum-copper alloys during ion beam etching. *Nuclear Instruments and Methods in Physics Research* B7:836–39.

Widom, E., M. S. Weathers, W. A. Bassett, D. L. Kohlstedt, and A. F. Anderson. (1984.) Microanalysis of glass inclusions in grain boundaries of Kilauea olivine aggregates. Paper read at fall meeting, American Geophysical Union, 2–8 December 1984, in San Francisco, CA.

Wolfenstine, J., D. Dimos, and *D. L. Kohlstedt.* 1985. Kinetic decomposition of Ni_2SiO_4. Paper read at spring meeting, American Chemical Society, 6–10 May 1985, in Cincinnati, OH.

Yamada, I., *C. J. Palmstrom,* E. Kennedy, *J. W. Mayer,* H. Inokawa, and T. Takagi. 1985. Epitaxy of aluminum films on semiconductors by ionized cluster beam. In *Layered structures, epitaxy and interfaces,* ed. J. M. Gibson and L. R. Dawson, pp. 3–21. Boston: D. Reidel.

Yang, A. C.-M., and *E. J. Kramer.* 1985. Craze fibril structure and coalescence by low angle electron diffraction. *Journal of Polymer Science* 23:1353–67.

Zheng, L. R., L. S. Hung, and *J. W. Mayer.* 1985. Redistribution of dopant arsenic during silicide formation. *Journal of Applied Physics* 58:1505–14.

■ MECHANICAL AND AEROSPACE ENGINEERING

Auer, P. L., M. Alonso, and J. Barkenbus. 1985. Prospects for commercial nuclear power and proliferation. In *The nuclear connection,* ed. A. Weinberg, M. Alonso, and J. N. Barkenbus, pp. 19–47. New York: Paragon House.

Dawson, P. R. (1984.) A model for the hot or warm working of metals with special use of deformation mechanism maps. *International Journal of Mechanical Sciences* 26(4):227–44.

Dawson, P. R., J. Lipkin, and H. S. Lauson. 1985. A state variable model for volumetric creep of clays. *ASCE Journal of Engineering Mechanics* 111(1):42–61.

Dewhurst, T. B., and *P. R. Dawson.* (1984.) Analysis of large plastic deformations at elevated temperatures using state variable constitutive models. In *Constitutive equations: Micro, macro and computational aspects,* pp. 149–64. Proceedings, annual winter meeting, American Society of Mechanical Engineers. New York: ASME.

Ettestadt, D., and *J. L. Lumley.* (1984.) Parameterization of turbulent transport in swirling flows: I. Theoretical considerations. In *Turbulent shear flows: 4,* ed. L. J. S. Bradbury, F. Durst, B. E. Launder, F. W. Schmidt, and J. H. Whitelaw, pp. 87–101. Berlin: Springer-Verlag.

Gates, R. S., N. R. Scott, R. E. Pitt, and *D. L. Bartel.* (1984.) Biomechanics of teat/liner interactions. Paper no. 84–3535, read at annual winter meeting, American Society of Agricultural Engineers, 11–14 December 1984, in New Orleans, LA.

George, A. R., and *S.-T. Chou.* 1985*a*. A comparative study of tail rotor noise mechanisms. Paper read at 41st Annual Forum, American Helicopter Society, 15–17 May 1985, in Fort Worth, TX.

———. 1985*b*. Broadband noise of propellers and rotors. In *Proceedings, NOISE-CON 85,* pp. 461–68. New York: The Noise Control Foundation.

Gouldin, F. C., J. S. Depsky, and S.-L. Lee. 1985. Velocity field characteristics of a swirling flow combustor. *AIAA Journal* 23:95–102.

Hollis, P., and *D. Taylor.* (1984.) Hopf bifurcation in multi-degree-of-freedom systems using MACSYMA. In *Proceedings of MACSYMA Users' Conference,* pp. 169–85. Cambridge, MA: MIT Laboratory for Computer Science.

Kwon, T. H., D. Chu, and *K. K. Wang.* 1985. Pressure loss prediction in a runner-gate-cavity system including juncture loss. In *Proceedings, ANTEC '85,* pp. 809–13. Society of Plastics Engineers Technical Papers, vol. 31. Stamford, CT:SPE.

Kwon, T. H., and *S. F. Shen.* (1984.) Application of recently proposed constitutive model for polymeric liquids to stress relaxations after various shear-deformation histories. In *Proceedings, 9th International Congress on Rheology,* ed. B. Mena, A. García-Rejón and C. Rangel-Nafaile, vol. 2, pp. 347–54. México: Universidad Nacional Autónoma de México.

Leibovich, S. 1985. Dynamics of Langmuir circulations in a stratified ocean. In *The ocean surface,* ed. Y. Toba and H. Mitsuyasu, pp. 457–64. Dordrecht, Netherlands: Reidel.

Leibovich, S., and *S. Lele.* 1985. The influence of the horizontal component of Earth's angular velocity on the instability of the Ekman layer. *Journal of Fluid Mechanics* 150:41–87.

Leu, M. C. (1984.) Solid geometric modeling towards robot intelligence. In *NSF study on supercomputers in mechanical systems research,* pp. MSXVIII-1–MSXVIII-4. Livermore, CA: Lawrence Livermore National Laboratory.

Leu, M. C., and M. Jirapongphan. 1985. Modeling and analysis of flow-induced vibrations in circular saws. *Journal of Vibration, Acoustics, Stress, and Reliability in Design* 107:196–202.

Leu, M. C., and *R. Mahajan.* (1984.) Computer graphic simulation of robot kinematics and dynamics. In *Proceedings, Robots 8 Conference,* pp. 4.80–4.101. Dearborn, MI: Society of Manufacturing Engineers.

Leu, M. C., and *S. H. Park.* (1984.) Application of PADL-2 solid modeler to robot simulation. In *Proceedings, 3rd Canadian CAD/CAM and Robotics Conference,* pp. 10.13–10.19. Ancaster, Ontario: Canadian Institute of Metalworking.

Lin, G. C. I., and *M. C. Leu.* (1984.) The role of dynamic computer graphics simulation in the use of robots. Paper read at National Conference and Exhibition on Robotics, 20–24 August 1984, in Melbourne, Australia.

Lumley, J. L., and I. Van Cruyningen. 1985. Limitations of second order modeling of passive scalar diffusion. In *Frontiers in fluid mechanics,* ed. S. H. David and *J. L. Lumley,* pp. 199–218. Berlin: Springer-Verlag.

Moore, F. K., and E. M. Greitzer. 1985. A theory of post-stall transients in axial compression systems: I. Development of equations; II. Application. Papers 85-GT-171 and 85-GT-172 read at 30th International Gas Turbine Conference and Exhibit, American Society of Mechanical Engineers, 18–21 March 1985, in Houston, TX.

Shmilovich, A., and *D. A. Caughey.* 1985. Calculation of transonic potential flow past wing-tail-fuselage configurations using the multi-grid method. In *Proceedings, 9th International Conference on Numerical Methods in Fluid Dynamics,* pp. 508–13. Lecture notes in physics, vol. 218. Berlin: Springer-Verlag.

Tseng, H. C., C. A. Hieber, K. K. Wang, H. H. Chiang, and *G. E. Grant.* 1985. Analysis of rheological data from an automated-injection-molding capillary rheometer. In *Proceedings, ANTEC 85,* pp. 716–19. Society of Plastics Engineers technical papers, vol. 31. Stamford, CT: SPE.

Wang, V. W., C. A. Hieber, and *K. K. Wang.* 1985. An interactive simulation of cavity filling in injection molding with color graphics. In *Proceedings of ANTEC 85,* pp. 826–29. Society of Plastics Engineers technical papers, vol. 31. Stamford, CT: SPE.

Wright, T. M., and *D. L. Bartel.* (1984.) Surface damage in polyethylene total knee components. In *Advances in bioengineering,* pp. 102–03. New York: American Society of Mechanical Engineers.

■ OPERATIONS RESEARCH AND INDUSTRIAL ENGINEERING

Avram, F., and *M. S. Taqqu.* 1985. Generalized powers of strongly dependent random variables. In *Seminar notes on multiple stochastic integration, polynomial chaos and their applications,* ed. W. A. Woyczynski. Cleveland, OH: Case Western Reserve University.

Bechhofer, R. E. and A. C. Tamhane. 1985. Tables of admissible and optimal balanced treatment incomplete block designs. *Selected tables in mathematical statistics* 8:41–139.

Bechhofer, R. E., and *D. M. Goldsman.* 1985*a*. On the Ramey-Alam seqeuntial procedure for selecting the multinomial event which has the largest probability. *Communications in Statistics—Simulation and Computation* 14(2):263–82.

———. 1985*b*. Truncation of the Bechhofer-Kiefer-Sobel sequential procedure for selecting the multinomial event which has the largest probability. *Communications in Statistics—Simulation and Computation* 14(2):283–315.

Boudouris, J., and *B. W. Turnbull.* 1985. Shock probation in Iowa. *Journal of Offender Counseling, Services, and Rehabilitation* 9:53–67.

Fox, R., and *M. S. Taqqu.* 1985. Non-central limit theorems for quadratic forms in random variables having long-range dependence. *Annals of Probability* 13:428–46.

Jackson, P. L. (1984.) Debt vs. equity in a simple model with transaction costs: I. Form of the optimal policy. *Operations Research Letters* 3(5):251–56.

Jackson, P. L., W. L. Maxwell, and *J. A. Muckstadt.* 1985. The joint replenishment problem with a powers-of-two restriction. *IIE Transactions* 17(1):25–32.

Prabhu, N. U. (1984.) Wiener-Hopf factorization of Markov semigroups: I. The countable state space case. In *Proceedings, 7th Conference on Probability Theory,* ed. M. Iosifescu, pp. 315–24. Bucharest: Editura Academiei Republicii Socialiste Romania.

Prabhu, N. U., and P. K. Reeser. 1985. A random family of queuing systems with a dynamic priority discipline. *Mathematics of Operations Research* 10:24–32.

Taqqu, M. S. 1985. Orthogonal processes. In *Encyclopedia of statistical sciences,* ed. S. Kotz and N. Johnson, pp. 536–37. New York: Wiley.

Taqqu, M. S., and *C. Czado.* 1985. A survey of functional laws of the integrated logarithm for self-similar processes. *Stochastic models* 1:77–155.

Taqqu, M. S., and *Fox, R.* 1985. Non-central limit theorems for quadratic forms in random variables having long-range dependence. *Annals of Probability* 13:428–46.

Todd, M. J. 1985. "Fat" triangulations, or solving certain non-convex matrix optimization problems. *Mathematical Programming* 31:123–36.

Weiss, L. 1985. Tests of independence of continuous random variables which guard against special alternatives. *Naval Research Logistics Quarterly* 32:337–46.

■ THEORETICAL AND APPLIED MECHANICS

Burns, J. A. (1984.) Planetary rings. In *Advances in space research,* vol. 4, ed. G. Morfill, C. T. Russell, and M. S. Hanner, pp. 121–34. London: Pergamon.

Chandra, A., and *S. Mukherjee.* (1984.) A finite element analysis of metal forming processes with thermomechanical coupling. *International Journal of Mechanical Sciences* 26:661–76.

Hart, E. W., and *Y.-W. Chang.* 1985. Material rotation effects in tension-torsion testing: Experimental results. In *Plasticity today: Modelling, methods, and applications,* ed. A. Sawczuk and G. Bianchi, pp. 235–46. New York: Elsevier.

Holmes, P. J. 1985*a*. Dynamics of a nonlinear oscillator with feedback control: I. Local analysis. *ASME Journal of Dynamic Systems, Measurement and Control* 107:159–65.

———. 1985*b*. Review of *The Lorenz equations: Bifurcations, chaos, and strange attractors,* by C. Sparrow. *SIAM Review* 27:106–10.

Schaffer, L., and *J. A. Burns.* (1984.) Dust motion in Jupiter's tilted magnetic field. In *Advances in space research,* vol. 4, ed. G. Morfill, C. T. Russell, and M. S. Hanner, pp. 107–10. London: Pergamon.

LETTERS

The Rise of Research

Editor: Julian C. Smith ["The Rise of Research: Fifty Years of Change in Cornell's Engineering Program" in the Summer 1985 issue of the *Quarterly*] should realize that students in the 1930s were unaware of the "decline in reputation" of the College. They were acutely aware, however, that "the College was overwhelmingly a teaching institution, with emphasis on training young men for careers in the rough-and-ready world of professional engineering practice."

The College of Engineering in the 1930s was certainly not worse off than other prominent schools. In those early post-depression years, we students were fortunate to be at Cornell and worked hard to prepare for the "rough-and-ready" practice of engineering. I was proud to receive at graduation in 1935 a civil engineering degree (not a B.S. in C.E.) in recognition of the caliber of the curriculum at that time. Our reputation then, as perhaps now, as a top College of Engineering was freely acknowledged by our peers in professional engineering practice.

As for our teaching faculty in the 1930s, Professor Smith makes it abundantly clear that our academic work was practical in orientation, taught by professors and assistant professors without Ph.D.s, who did little research and published even less....We alumni remember them fondly and, over the years, have appreciated more and more what they did to provide us with a first-class engineering education. [Professor Smith should keep this in mind while] raising so high the banner for research and plotting on probability paper the percent of professors with Ph.D.s.

Just four years before Hitler's invasion of Poland, the engineering graduates in 1935 were unknowingly being trained to serve our country in World War II. Those who survived performed outstandingly in the years following, improving the welfare of mankind....Today's College of Engineering will do well to...graduate students as well prepared to tackle the real world of engineering.

Haywood G. Dewey, Jr. '35
Sacramento, California

Reply from Julian C. Smith: In preparing my article, I tried to chronicle the changes that have occurred in the College of Engineering over the past fifty years, especially in attitudes toward research, but to avoid value judgements as much as I could. Certainly there was no intention of giving an uncharitable picture of the faculty and students of the 1930s. I was a Cornell student then myself, and I also was not aware of the long period of decline in the College between 1910 and 1935. I too admired the many distinguished faculty members and felt the training I received was excellent preparation for my work in industry during World War II. And now, in my fortieth year of teaching at Cornell, I wouldn't say that our current engineering students are any smarter or better prepared for engineering practice than they were in those days.

But engineering practice is different, and the subject matter is different, and the faculty's attitude toward research and publication is very different. About a third of our engineering students are graduate students; about a third are women. Research is big business in the College. The history of these changes, and their underlying causes, were the subjects of my article. I tried to report them accurately, not to judge them.

As to the plot of the percentage of doctor's degrees in the engineering faculty, I found it amusing and rather surprising that it made such a good straight line on probability paper—almost as if some external force were driving the percentage upward. Is it good that almost all the faculty members—96.3 percent—now have doctor's degrees? I don't know. I don't have one myself—my degree is Chemical Engineer. A doctorate doesn't necessarily make a good engineer, or make him or her unfit for engineering practice, but it is absolutely essential to getting a teaching position in any leading engineering college. That's just the way things are these days.

The Fear of Pollutants

Editor: Professor Neil Orloff's Commentary in the Summer 1985 issue, "Is the Fear of Pollutants Overemphasized in Our Society?" focuses on people's perceptions of risks. He states that in the U.S. one of our greatest fears is of toxic chemicals (although there are greater risks of mortality from other sources such as car accidents and mountain climbing), and points out that people's focus

"reflects their view of moral and immoral conduct" and that dangers perceived "are exaggerated or minimized according to the social acceptability of the underlying activities." He then suggests three reasons why there is such widespread concern about chemicals: (1) the presence of chemicals in, for example, foods, provides a simple and plausible explanation for development of a dreaded disease such as cancer—and thus absolves the individual from responsibility; (2) it is simpler and [perversely?] more reassuring to blame man-made chemicals rather than nature; and (3) a focus on chemicals provides an outlet for anti-business attitudes, such as the "power and perceived indifference of large corporations." Dr. Orloff believes that because [regulatory?] "officials have failed to consider the[se] cultural factors underlying people's fear of chemicals", they could not possibly have properly approached environmental controversies, since all they have to offer the fearful public is generalized risk statements about, for example, only a one-in-a-million chance of cancer (a small risk).

Dr. Orloff short-changes his readers by failing to offer any evidence supporting such colorful suggestions as being the basis of widespread public concern and by neglecting at least two other relevant aspects of the discussion. First, he ignores what I and many others believe to be one of the most important characteristics of risk perception: its "voluntariness" (that is, is the risk assumed voluntarily, or is it imposed on one?). Second, his article omits any real consideration of the "track record" of toxic chemicals and "high tech"—for example, how often and how repeatedly we have been unpleasantly suprised, not only by a well-known chemical pesticide like DDT, but also by such practices as the formerly routine x-raying of children's feet to see how their shoes fit, or the routine use of food additives such as dyes, which were once "generally regarded as safe" (GRAS) and used without adequate testing.

I'm not at all sure (as one can infer Dr. Orloff to be) that an individual would refuse to take responsibility for his chemical-demanding/chemical-based lifestyle—if he knew about it. But there is no indication at the federal administratiave level or national industry level of any interest in helping the consumer make educated, informed choices. In fact, there is no indication of any attempt to view things from the consumer's point of view at all—for example, to require comprehensive food and beverage and material labeling; to point out what is known and not known; to support the Delaney clause; to forbid overseas sale of products banned in the U.S.—in other words, to undertake honest public education and to demonstrate that the consumer should believe what regulatory agencies, government, and business tell him about risk. These major players on the scene have lost the trust of the lay person and must earn it back. There is no simple prescription for this—except to remember that actions speak louder than words.

Thus, it is not so much that people are irrational to fear something they are told has a low level or risk as they are wise to have learned from history that such statements have been wrong before. That historical "knowledge" is reflected in their attitudes toward accepting involuntary risk, whether chemical, nuclear, or otherwise.

Dooley Kiefer, A.B. '57
Ithaca, New York

Architectural History

Editor: I recognize that the future direction of the College is more important than the real-estate history; however, I wish to offer corrections to several captions accompanying Kermit C. Parson's article on the history of the engineering facilities at Cornell. This appeared in the summer 1985 issue, "Building for a New Era."

On page 20, the caption for a 1936 aerial photograph of the main Cornell campus indicates that the only building on the site of the present engineering quadrangle was an armory and gymnasium. Actually, the site was occupied by faculty residences from the location of Kimball-Thurston through the locations of Upson and Phillips. These faculty residences are readily visible in the oblique aerial view. Also, Kappa Alpha is barely visible over the roof of Myron Taylor. The 1948 proposal on page 21 indicates Kappa Alpha as K.A.

Faculty residences, and the armory and gymnasium are mentioned in the page 14 caption pertaining to the 1902 proposal shown on page 15. However, there are two other buildings on the map. The building southwest of the armory is Kappa Alpha. The building southeast of the armory is a former central heating plant. The armory and gymnasium, which are the same building, stood until 1956; the projection to the south enclosed a swimming pool. Kappa Alpha was occupied at this location at least until 1957.

Richard P. Spiro, BCE '59
Bronx, New York

Editor's Note: Another error in the Summer issue (on page 25) is a misspelling of the name of the architect for Snee Hall. He is Mario L. Schack, who was then in the Washington office of Perkins & Will. Before joining that firm, Schack was chairman of Cornell's architecture department, and he has now returned to the University as a professor of architecture in the College of Architecture, Art, and Planning.

K. C. Parsons: I've just read your "Building for Engineering at Cornell" in the *Quarterly*. It's a fine history, and of interest to college and university planners generally. Also, the P&W staff will enjoy seeing the good coverage of Snee Hall.

Bill Brubaker, Senior Partner
Perkins & Will
Chicago, Illinois

An Editor's Testimonial

Computer science affects the working methods of practically everyone these days, and we are no exception. The ways in which publications are written, edited, and produced have changed dramatically in the past few years. We are always glad to tell about what has been happening to *Engineering: Cornell Quarterly* and other College publications, and this issue on computer science provides a good opportunity.

As recently as the beginning of 1982, the *Quarterly* was edited in the red-pencil mode, with instructions for typesetting written in the margins of the copy. Proofs were received in long *galleys*, corrections were written in, and the galleys were returned to the typesetter along with a dummy layout. Next the printer prepared a *mechanical*, which was photographed to make plates for the actual printing. All this took about two months from the time the copy was ready.

The first change in our office was the introduction of a word processor, which facilitated writing and editing. (We also use it to maintain our subscription files and to print mailing labels.) Then, using a modem, we began to transmit copy over the telephone lines to our printer in Massachusetts; carefully-worked-out codes replaced hand-written "specs" and the copy was set automatically without having to be retyped. For the issue before this one, we began to use a local typesetter who could supply proofs the next day, and to prepare the mechanicals ourselves. Small changes could be taken care of right away, simply by transmitting a few new lines and pasting in the resulting patches on the mechanical. Weeks of "turn-around" time were saved and production costs were reduced by about $10,000 a year.

We began editing in the days when type was set in lead on huge clanking Linotype machines. Innovations in printing techniques changed that and much more, but until quite recently, editors still worked with the traditional sequence of typed copy, galley proofs, page proofs, and bluelines. Now computer science and its sister disciplines provide new tools almost faster than we can keep up with them. For instance, in the article in this issue by Tim Teitelbaum and Thomas Reps, we read about a WYSIWYG system that eliminates visible coding from word-processor output so that What You See Is What You Get; with this system, we would be able to code manuscripts as we input them and still be able to print out readable copies for authors' approval. With improved interfacing, we could eliminate some retyping by copying an author's diskette and making editing changes directly on it. Maybe some day soon we can simplify mechanical preparation by formatting right on the screen.

Computerized publication doesn't improve content, of course, but it enables an editor to be more productive and decrease costs. Here's a vote of confidence for computer technology from a happy user.—GMcC

ENGINEERING
Cornell Quarterly

Published by the College of Engineering, Cornell University

Editor: Gladys McConkey

Associate Editor: David Price

Editorial Assistant: Lindy Costello

Graphic Art Work: Francis Russell

Typesetting: Davis Graphic Services
Lansing, New York

Printing: Davis Press, Inc.
Worcester, Massachusetts

Photography credits:

Jon Reis: 1 (top), 44 (top), 47 (left), 48, 49 (right)
David Ruether: inside front cover, 1 (except top), 10, 11, 17, 22, 32, 40, 43, 45, 47 (right), 49 (left)

Please address any correspondence, including notification of change of address, to *ENGINEERING: Cornell Quarterly,* Carpenter Hall, Ithaca, New York 14853-2201.

ENGINEERING
Cornell Quarterly
ISSN 0013-787X
Carpenter Hall, Ithaca, N.Y. 14853

Second Class
Postage Paid at
Ithaca, N.Y. 14850
and additional offices