Living with a New Mathematical Species

Computing and computer science prove again that mathematics is a living part of human culture.

by Lynn Arthur Steen

In 1985 the International Commission on Mathematical Instruction (ICMI) convened a meeting in France of mathematicians, computer scientists, and mathematics educators from around the world to discuss the impact of "informatics" (the common European term for computer science*) on mathematics—specifically, on mathematical research, on the mathematical curriculum, and on mathematical pedagogy.*

Prior to the 1985 conference, an ICMI Steering Committee prepared a draft paper on "the influence of computers and informatics on mathematics and its teaching." This paper was circulated worldwide to interested mathematics educators, and served as the focal point for the meeting.

This ABACUS *article is adapted from one of the responses to that ICMI discussion document. A formal report from the ICMI study, including several responses, has been published by Cambridge University Press* [The Influence of Computers and Informatics on Mathematics and Its Teaching].

In this article, a dagger [†] *indicates an expression defined in the glossary on page 41.*

Computers are both the creature and the creator of mathematics. They are, in the apt phrase of Seymour Papert, "mathematics-speaking beings." More recently, J. David Bolter, in his stimulating book *Turing's Man*, calls computers "embodied mathematics." Computers shape and enhance the power of mathematics, while mathematics shapes and enhances the power of computers. Each forces the other to grow and change, creating—in Thomas Kuhn's language—a new mathematical paradigm.

Until recently, mathematics was a strictly human endeavor. It evolved with human society, achieving a degree of universality equalled by few other aspects of human culture. Its ecology was a human ecology, linked closely to science and language, evolving as human science and language changed.

But suddenly, in a brief instant on the time scale of mathematics, a new species has entered the mathematical ecosystem. Computers speak mathematics, but in a dialect that is difficult for some humans to understand. Their number systems are finite rather than infinite; their addition is not commutative; and they don't really understand "zero," not to speak of "infinity." Nonetheless, they do embody mathematics.

Many features of the new computer mathematics appear superficial: notation such as ∧ and ** for exponentiation; linearized expressions for formulas traditionally represented by a two-dimensional layout; a preference for binary, octal, or hexadecimal representations of numbers; and in early languages, a new action-

Lynn Arthur Steen is a professor of mathematics at St. Olaf College in Northfield, Minnesota, and president of the Mathematical Association of America. He is editor or author of seven books, including *Mathematics Today* and *Mathematics Tomorrow*, a pair of volumes that survey the state of contemporary mathematics for the general reader. He has written numerous articles about mathematics and computer science for such periodicals as *Scientific American*, *Science News*, and *Science*, and for five years was coeditor of *Mathematics Magazine*. Steen is Secretary of Section A, Mathematics, of the American Association for the Advancement of Science, and is a member of the Executive Committee of the Mathematical Sciences Education Board of the National Research Council. He received a Ph.D. in mathematics from M.I.T. in 1965.

oriented meaning to the equals sign. Some variances are more significant, and more difficult to assimilate into traditional mathematics: finite number systems, interval arithmetic, roundoff errors, or computational intractability.

As mathematics goes, linguistic and notational changes are truly superficial—it really is the same subject modulo an isomorphism. These differences can be very confusing to students learning mathematics and computing, although perhaps no more so than the differences in vocabulary and perspective between an engineer and a mathematician. The blend of computer language and traditional mathematics produces a kind of Franglais, decried by purists yet employed by everyone.

The core of mathematics, however, is also changing under the ecological onslaught of mathematics-speaking computers. New specialties in computational complexity, theory of algorithms, graph theory, and formal logic attest to the impact that computing is having on mathematical research. As Harvard physicist Arthur Jaffe has argued so well in his recent essay "Ordering the Universe," the computer revolution *is* a mathematical revolution. The intruder has changed the ecosystem of mathematics, profoundly and permanently.

New Mathematics for a New Age

Computers are discrete, finite machines. Unlike a Turing machine with an infinite tape, real machines have limits of both time and space. Theirs is not an idealistic Platonic mathematics, but a mathematics of limited resources. The goal is not just to get a result, but to get the best result for the least effort. Optimization, efficiency, speed, productivity—these are essential objectives of modern computer mathematics. Questions of optimization lead to the study of graphs, of operations research, of computational complexity.

Computers are also logic machines. They embody the fundamental engine of mathematics—rigorous propositional calculus. So it comes as no surprise that computer programs can become full partners in the process of mathematical proof. The first celebrated computer proof was that of the four-color theorem: the computer served there as a sophisticated accountant, checking out thousands of cases of reductions. Despite philosophical alarms that computer-based proofs would change mathematics from an *a priori* to a contingent, fallible subject, careful analysis reveals that nothing much had really changed. The human practice of mathematics has always been fallible; now it has a partner in fallibility.

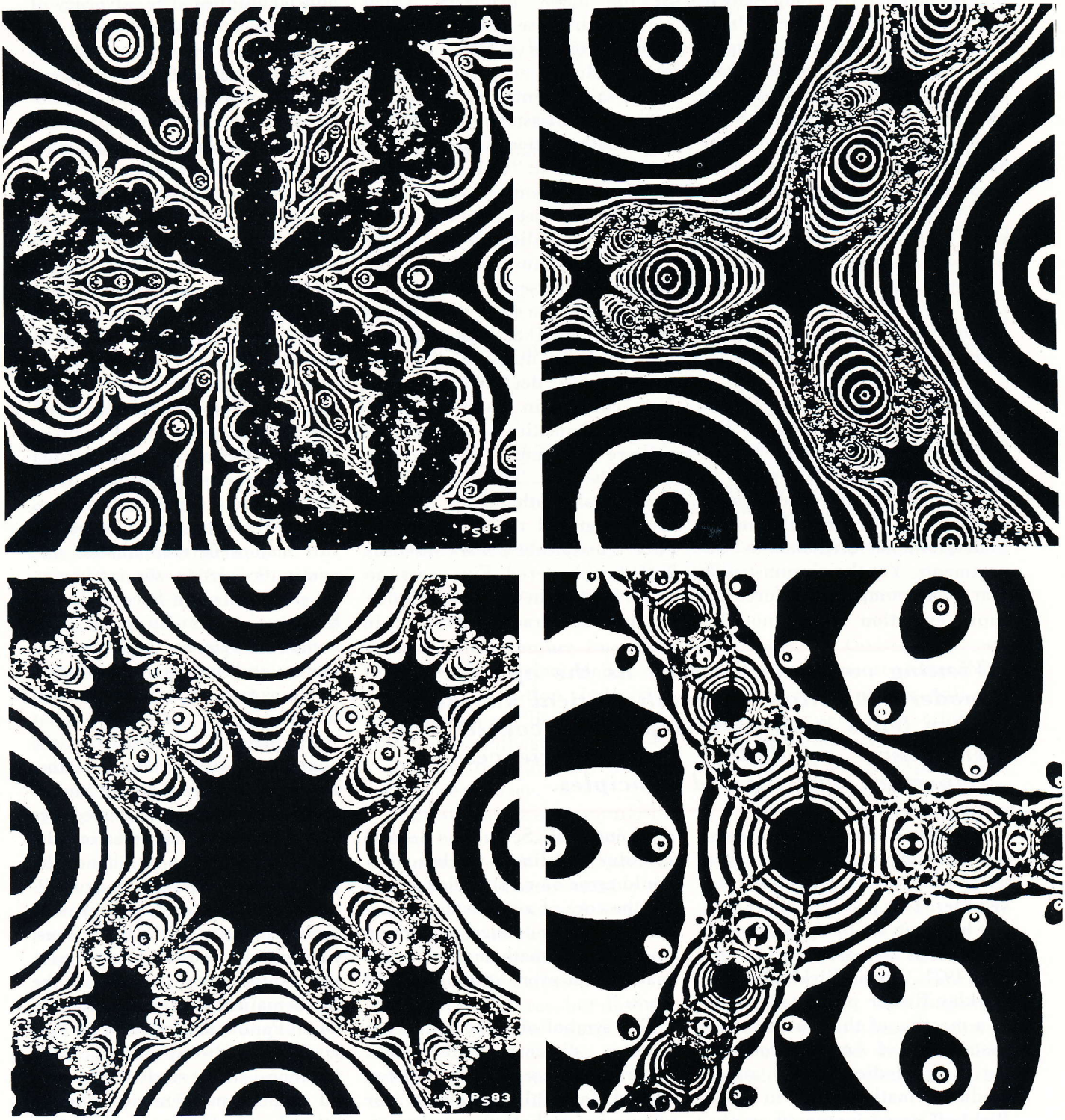
Recent work on the mysterious Feigenbaum constant† reveals just how far this evolution has progressed in just eight years: computer-assisted investigations of families of periodic maps suggested the presence of a mysterious universal limit, apparently independent of the particular family of maps. Subsequent theoretical investigations led to proofs that are true hybrids of classical analysis and computer programming: the crucial step in a fixed-point argument† requires a tight estimate on the norm of a high-degree polynomial. This estimate is made by a computer program, carefully crafted using interval arithmetic to account in advance for all possible inaccuracies introduced by roundoff error. Thus computer-assisted proofs are possible not just in graph theory, but also in that bastion of classical mathematics, functional analysis.

Computers are also computing machines. By absorbing, transforming, and summarizing massive quantities of data, computers can simulate reality. No longer need the scientist build an elaborate wind tunnel or a scale-model refinery in order to test engineering designs. Wherever basic science is well understood, computer models can emulate physical processes by carrying out instead the process implied by mathematical equations. Mathematical models used to be primarily tools used by theoretical scientists to formulate general theories; now they are practical tools of enormous value in the everyday world of engineering and economics. They focus mathematical attention on the relation between data and theory, on stochastic processes† and differential equations, on data analysis and mathematical statistics.

In many respects mathematics has become the creature of the computer: by providing compelling tools in combinatorics, logic, and calculation, the computer has made an offer of intellectual adventure that mathematicians cannot refuse. But in a very real sense, mathematics is also the creator of the computer. David Hilbert's struggle with the foundations of mathematics—itself precipitated by the paradoxes of set theory elucidated by Frege and Russell—led directly to Alan Turing's proposal for a universal machine of mathematics.

It has been fifty years since Turing developed his scheme of computability, in which he argued that machines could do whatever humans might hope to do. His was a formal, abstract system, devoid of hardware and real machines. It took two decades of engineering effort to turn Turing's abstractions into productive real machines.

During that same period, abstract mathematics flourished, led by Bourbaki, symbolized by the "generalized abstract nonsense" of category theory†. But with abstraction came power; with rigor



The fusion of mathematics with computer technology, along with a sense of the aesthetic, is embodied in the study of visual representations of Julia sets—work done by Heinz-Otto Peitgen and Dietmar Saupe, who prepared these graphics at the University of Utah. Three of the pictures are derived from the relaxed Newton method $N_\lambda(z) = z - \lambda\, p(z)/p'(z)$ for $p(z) = z^3 - 1$ in $[-1,1] \times [-1,1]$; clockwise from upper left, the patterns are based on λ = 1.5, λ = 1.0, and λ = 0.5, respectively. The lower left picture is a representation of Newton's Method for $z^4 - 1 = 0$. [For more information, see "Cayley's Problem and Julia Sets" by Peitgen, Saupe, and von Haeseler, *The Mathematical Intelligencer,* 6(2):11-20.]

came certainty. Once real computers emerged, the complexity of programs quickly overwhelmed the informal techniques of backyard programmers. Formal methods became *de rigueur*; even the once-maligned category theory is now enlisted to represent finite automata and recursive functions. Once again, as happened before with physics, mathematics became more efficacious by becoming more abstract.

Changing the Mathematics Curriculum

Twenty years ago, in the United States, the Committee on the Undergraduate Program in Mathematics (CUPM) issued a series of reports that led to a gradual standardization of curricula among undergraduate mathematics departments. Yet the circumstances that make computing a force for rapid evolution in the notation and practice of mathematics also put pressure on the mathematics curriculum in colleges and universities. This pressure is not new, but has been building in intensity throughout the past decade.

In 1971, Garrett Birkhoff and J. Barkley Rosser presented papers at a meeting of the Mathematical Association of America concerning their predictions for undergraduate mathematics in 1984. Birkhoff urged increased emphasis on modelling, numerical algebra, scientific computing, and discrete mathematics. He also advocated increased use of computer methods in pure mathematics: "Far from muddying the limpid waters of clear mathematical thinking, they make them more transparent by filtering out most of the messy drudgery which would otherwise accompany the working out of specific illustrations." Rosser emphasized many of the same points, and warned of impending disaster to undergraduate mathematics if their advice went unheeded: "Unless we revise [mathematics courses] so as to embody much use of computers, most of the clientele for these courses will instead be taking computer courses in 1984."

In the decade since these words were written, U.S. undergraduate and graduate degrees in mathematics have declined by 50%. New courses in modelling, discrete mathematics, and data analysis are emerging in every college and university. The clientele for traditional mathematics has indeed migrated to computer science. The former CUPM consensus is all but shattered. Five years ago CUPM issued a new report, this one on the Undergraduate Program in Mathematical Sciences. Beyond calculus and linear algebra, they could agree on no specific content for the core of a mathematics major: "There is no longer a common body of pure mathematical information that every student should know."

Viewing computer literacy as the appropriate modern substitute for mathematical knowledge often leads students to superficial courses that emphasize vocabulary and experiences over important concepts and principles.

The symbol of reformation has become discrete mathematics. Several years ago Anthony Ralston argued forcefully the need for change before both the mathematics community and the computer science community. Discrete mathematics, in Ralston's view, is the central link between the fields. College mathematics must introduce discrete methods early and in depth; computer science curricula must, in turn, require and utilize mathematical concepts and techniques. The advocacy of discrete mathematics rapidly became quite vigorous, and the Sloan Foundation funded experimental curricula at six institutions to encourage development of discrete-based alternatives to standard freshman calculus.

The niche of mathematics in the university ecosystem has been radically transformed by the presence of computer science in the undergraduate curriculum. The strongest mathematics departments continue to offer the traditional CUPM major, oftentimes for a declining number of students. Many smaller departments, however, have been forced to drop regular offerings of such former core courses as topology, analysis, and algebra. In such institutions, where resources do not permit full majors in mathematics and computer science, the mathematics program often becomes a hybrid major consisting of some computer science, some mathematics, and some statistics—introductions to everything, mastery of nothing.

The need for consensus on the contents of undergraduate mathematics is perhaps the most important issue facing American college and university mathematics departments. On the one hand, departments that have a strong traditional major often fail to provide their students with the robust background required to survive the evolutionary turmoil in the mathematical sciences. Like the Giant Panda, they depend for survival on a dwindling supply of bamboo—strong students interested in pure mathematics. On the other hand, departments offering flabby composite majors run a different risk: by avoiding advanced, abstract requirements, they often misrepresent the true source of mathematical knowledge and power. Like zoo-bred animals unable to forage in the wild, students who have never

Glossary

category theory	A general theory of mathematical structures, emphasizing similarity of form by means of mappings (called *functors*) from one structure to another.
Feigenbaum constant	The number 4.669196223 . . . , calculated as the limit of the ratio of differences between successive values of a parameter α at bifurcation points of iterations of a map $x \rightarrow f_\alpha(x)$. This constant arises experimentally in many contexts; M. Feigenbaum showed that it is independent of the particular family of functions involved in the iteration.
fixed-point argument	A method of proof derived from topology which, under certain circumstances, guarantees a solution to an equation of the form $f(x) = x$.
Ising model	A simple model for magnetization strength in permanent magnets based on tiny abstract magnets or "spins" arranged in a regular lattice.
stochastic process	A sequence or continual process of random events.

been required to master a deep theorem are ill-equipped to master the significant theoretical complications of real-world computing and mathematics.

Computer Literacy

Mathematical scientists at American institutions of higher education are responsible not only for the technical training of future scientists and engineers, but also for the quantitative literacy of lay people—of future lawyers, politicians, doctors, educators, and clergy. Public demand that college graduates be prepared to live and work in a computer age has caused many institutions to introduce requirements in quantitative or computer literacy. Many educators are calling for a total reform of liberal education.

In 1981 the Alfred P. Sloan foundation initiated curricular exploration of "the new liberal arts," the role of applied mathematical and computer sciences in the education of students outside technical fields: "The ability to cast one's thoughts in a form that makes possible mathematical manipulation and to perform that manipulation [has] become essential in higher education, and above all in liberal education." In November 1982, University of California President David Saxon wrote in a *Science* editorial that liberal education "will continue to be a failed idea as long as our students are shut out from, or only superficially acquainted with, knowledge of the kinds of questions science can answer and those it cannot."

Too often these days the general public views computer literacy as the appropriate modern substitute for mathematical knowledge. Unfortunately, this often leads students to superficial courses that emphasize vocabulary and experiences over concepts and principles. The advocates of computer literacy conjure images of an electronic society dominated by the information industries. Their slogan of "literacy" echoes traditional educational values, conferring the aura but not the logic of legitimacy.

Typical courses in computer literacy, however, are filled with ephemeral details whose intellectual life will barely survive the students' school years. A best-selling textbook in the United States for courses introducing computing to nonspecialists is full of glossy color pictures, but does not even mention the word "algorithm." These courses contain neither a Shakespeare nor a Newton, neither a Faulkner nor a Darwin; they convey no fundamental principles nor enduring truths.

Computer literacy is more like driver education than like calculus. It teaches students the prevailing rules of the road concerning computers: how to create and save files, how to use word processors and spreadsheets, how to program in Basic. One can be confident only that most students finishing such a course will not injure themselves or others in their first encounter with a real computer in the workplace. But such courses do not leave students well prepared for a lifetime of work in the information age.

Algorithms and data structures are to computer science what functions and matrices are to mathematics. As much of the traditional mathematics curriculum is devoted to elementary functions and matrices, so beginning courses in computing—by whatever name—should stress standard algorithms and typical data structures.

For example, as early as students study linear equations they could also learn about stacks and queues; when they move on to conic sections and quadratic equations, they could in a parallel course investigate linked lists and binary trees. The algorithms for sorting and searching, while not

part of traditional mathematics, convey the power of abstract ideas in diverse applications every bit as much as do conic sections or derivatives.

Computer languages can (and should) be studied for the concepts they represent—procedures in Pascal, recursion and lists for Lisp—rather than for the syntactic details of semicolons and line numbers. They should not be undersold as mere technical devices for encoding problems for a dumb machine, nor oversold as exemplars of a new form of literacy. Computer languages are not modern equivalents of Latin or French; they do not deal in nuance and emotion, nor are they capable of persuasion, conviction, or humor. Although computer languages do represent a new and powerful way to think about problems, they are not a new form of literacy.

Computer Science

The confusion evident in university mathematics departments is an order of magnitude less severe than that which operates in university computer science programs. In the United States, these programs cover an enormous spectrum, from business-oriented data-processing curricula through management information science to theoretical computer science. All of these intersect with the mathematics curriculum, each in different ways. The computer science community is now struggling with this chaos, and has a process in place for identifying exemplary programs of different types as a first step towards an accreditation system for college computer science departments.

Several computer science curricula have been developed by the professional societies ACM and IEEE, for both large universities and small colleges. Recently Mary Shaw of Carnegie-Mellon University put together a composite report on the undergraduate computer science curriculum [see BOOK REVIEWS in this issue, page 48]. This report is quite forceful about the contribution mathematics makes to the study of computer science: "The most important contribution a mathematics curriculum can make to computer science is the one least likely to be encapsulated as an individual course: a deep appreciation of the modes of thought that characterize mathematics."

The converse is equally true: one of the more important contributions that computer science can make to the study of mathematics is to develop in students an appreciation for the power of abstract methods when applied to concrete situations. Students of traditional mathematics used to study a subject called "Real and Abstract Analysis"; students of computer science now can take a course titled "Real and Abstract Machines." In the former "new math," as well as in modern algebra, students learned about relations, abstract versions of functions; today, business students study "relational data structures" in their computer courses, and advertisers tout "fully relational" as the latest innovation in business software. The abstract theories of finite-state machines and deterministic automata are reflections in the mirror of computer science of well-established mathematical structures from abstract algebra and mathematical logic.

An interesting and pedagogically attractive example of the power of abstraction made concrete can be seen in the popular electronic spreadsheets that are marketed under such trade names as Lotus and VisiCalc. Originally designed for accounting, they can equally well emulate cellular automata or the Ising model for ferromagnetic materials†. They can also be "programmed" to carry out most standard mathematical algorithms: the Euclidean algorithm, the simplex method, Euler's method for solving differential equations. An electronic spreadsheet—the archetype of applied computing—is a structured form for recursive procedures—the fundamental tool of algorithmic mathematics. It is, to echo David Bolter, mathematics embodied in a computer.

Computers in the Classroom

Computers are mathematics machines, as calculators are arithmetic machines. Just as the introduction of calculators upset the comfortable paradigm of primary-school arithmetic, so the spread of sophisticated computers will upset the centuries-old-tradition of college and university mathematics. This year, long division is passe; next year, integration will be under attack.

Reactions to machines in the mathematics classroom are entirely predictable. Committee oracles and curriculum visionaries proclaim a utopia in which students concentrate on problem solving and machines perform the mindless calculations (long division and integration). Yet many teachers, secure in their authoritarian rule-dominated world, banish calculators (and computers) from ordinary mathematics instruction, using them if at all for separate curricular units where different ground rules apply. The recent International Assessment of Mathematics documented that in the United States calculators are never permitted in one-third of the 8th grade classes, and rarely used in all but 5% of the rest.

The large gap between theory and practice in the use of computers and calculators for mathematics instruction is due in part to a pedagogical assumption that pits teacher against machine. If the teacher's role is to help (or force) students to learn the rules of arithmetic (or calculus), then any

machine that makes such learning unnecessary is more a threat than an aid. Debates continue without end: Should calculators be used on exams? Should we expect less mastery of complex algorithms like long division or integration? Will diminished practice with computation undermine subsequent courses that require these skills?

The impact of computing on secondary-school mathematics has been the subject of many recent discussions in the United States. University of Maryland mathematician Jim Fey, coordinator of two of the most recent assessments, described these efforts as "an unequivocal dissent from the spirit and substance of efforts to improve school mathematics that seek broad agreement on conservative curricula. Many mathematics educators working with emerging electronic technology see neither stability nor consensus in the future of school mathematics."

The technology wars are just beginning to spread to the college classroom. Lap-size computers are now common—they cost about as much as ten textbooks, but take up only the space of one. Herb Wilf, editor-elect of the *American Mathematical Monthly*, argues that it is only a matter of time before students will carry with them a device to perform all the algorithms of undergraduate mathematics. A recent survey of applied research based on symbolic algebra asserts that "it will not be long before computer algebra is as common to engineering students as the now obsolete slide rule once was."

John Kemeny tells a story about calculus instruction that sheds interesting new light on the debate about manipulating symbols. He asks for the value of $\int_0^{13} e^x \, dx$. A moment's thought reveals the answer to be $e^{13} - 1$. That's the exact answer. Kemeny's first question is this: What is its value to *one* significant digit? With just paper and pencil, that's hard to do—beyond the likely skills of typical calculus students. (The answer: 400,000.) Now comes the second question: What's the difference between the original question and the traditional exact answer? They are both exact expressions for the value we seek, equally unenlight-

Symbolic Computation

```
         Vaxima 2.04
       Sat Oct 12 00:34:17 1985
    [load /user/vaxima/vaximarc.l]

    (c1)
    (c2)
                                             1
    (d2)               f : ----------------------------------------
                                      3             2          2
                               (x - 4)  (x - 1) (x  + 1) (x  + x + 1)

    (c3)
    (c4)
    (d4)                             expand(f)

    (c5)
                                           1
    (d5)       --------------------------------------------------------------
                8       7       6       5       4        3       2
               x  - 12 x  + 49 x  - 77 x  + 60 x  - 113 x  + 76 x  - 48 x + 64

    (c6)
    (c7)
    (d7)                           integrate(f, x)

    (c8)
                                                      2 x + 1
                   2                2          20 atan(-------)
           log(x  + x + 1)   5 log(x  + 1)            sqrt(3)      99 atan(x)
    (d8)   --------------- - ------------- - ---------------- + ----------
                 343             19652         3087 sqrt(3)        9826
                 log(x - 1)   116131 log(x - 4)            880 x - 3877
               - ---------- + ----------------- + ---------------------------------
                    162           136497879                   2
                                                       764694 x  - 6117552 x + 12235104

    (c9)
    (c10)
    (d10)                          diff(%, x)

    (c11)
                      40                               880
    (d11) - ------------------------ + ---------------------------------
                            2                     2
                   (2 x + 1)              764694 x  - 6117552 x + 12235104
          9261 (------------ + 1)
                        3

       (880 x - 3877) (1529388 x - 6117552)          2 x + 1               5 x
     - ------------------------------------- + ----------------- - --------------
                   2                        2         2                     2
       (764694 x  - 6117552 x + 12235104)     343 (x  + x + 1)     9826 (x  + 1)
           99             1               116131
     + ----------- - ------------ + -------------------
              2      162 (x - 1)    136497879 (x - 4)
       9826 (x  + 1)
    (c12)
    (c13)
    (d13)                          ratsimp(%)

    (c14)
                                           1
    (d14)      --------------------------------------------------------------
                8       7       6       5       4        3       2
               x  - 12 x  + 49 x  - 77 x  + 60 x  - 113 x  + 76 x  - 48 x + 64

    (c15)
    (c16)
    (d16)                          factor(%)

    (c17)
                                          1
    (d17)               ----------------------------------------
                                3             2          2
                         (x - 4)  (x - 1) (x  + 1) (x  + x + 1)

    (c18)
```

Symbolic computation, as in this example from Macsyma, is increasingly important in many pure and applied branches of mathematics. Here, operations are performed on the function *f* (*x*) seen at the top.

ening. So the proper question is not to find an exact value, but to choose which of many possible exact values is more suitable to the purpose at hand.

The challenges of computers in the classroom are exactly analogous to those of calculators. The computer will do for the teaching of calculus algorithms just what calculators did for arithmetic computations—it will make them redundant. In so doing, it will challenge rigid teachers to find new ways to reassert authority. Good teachers, however, should respond to the computer as a blessing in disguise—as a *deus ex machina* to rescue teaching from the morass of rules and templates that generations of texts and tests have produced.

As mathematics and computer science interact, beginning students come to realize that mathematics is continually created in response to new internal and external challenges.

Following the Rules

Mathematics students, just like any other students, like to get correct answers. Computers, for the most part, reinforce the student's desire for answers. Their school uses have been largely extensions of the old "teaching machines": programmed drill with predetermined branches for all possible answers, right or wrong. In colleges and universities, computers are still used most often as black-box calculators, spewing out numbers in answer to questions both asked and unasked.

Core mathematics courses continue this long-standing tradition, reinforcing the emphasis on rules and answers. Traditional calculus textbooks bear an uncanny resemblance to the first calculus text ever published: l'Hopital's 1699 classic. They present rules of differentiation and integration, with or without proof: linearity, product and quotient rules, chain rule, substitution, and so forth. After each rule are exercises to practice on. At the end of the chapter are mixed exercises, where the challenge is to use all the rules at the same time.

Most students of even modest ability can master these rules. If there is one thing that school does well, it is training students to learn rules. Strong students master them quickly, and yearn for tough problems that extend the rules (for instance, to x^x). Weak students work by rote, carefully adhering to template examples. Students of all types flounder when presented with "word problems" to "apply" their skills on: "A farmer has 200 meters of fence with which to . . . ." Too often, such problems are merely mathematical crossword puzzles—stylized enigmas whose solutions depend in large part on recognizing the unstated problem pattern. Indeed, recent research in problem solving suggests that many students learn to solve such problems by establishing mental categories of problem-type, and of course many instructors teach students to identify such types.

The confluence of research on learning with symbolic algebra has produced a rich new territory for imaginative pedagogy. Symbolic algebra packages linked to so-called "expert systems" on computers of sufficient power (with high-resolution graphics, mouse-like pointers, and multiple windows) can provide an effective intelligent tutor for learning algebraic skills. Computers can manipulate algebraic and numerical expressions as well as students can, usually better. However, they cannot recognize, parse, or model a word problem except in the narrowest sense—by matching templates to canonical forms.

It is commonplace now to debate the value of teaching skills such as differentiation that computers can do as well or better than humans. Is it really worth spending one month of every year teaching half of a country's 18-year-old students how to imitate a computer? What is not yet so common is to examine critically the effect of applying to mathematics pedagogy computer systems that are only capable of following rules or matching templates. Is it really worth the time and resources to devise sophisticated computer systems for efficiently teaching precisely those skills that computers can do better than humans, particularly those skills that make the computer tutor possible? The basic question is this: since computers can now do algebra and calculus algorithms, should we use this power to reduce the curricular emphasis on calculations, or as a means of teaching calculations more efficiently? This is a new question, with a very old answer.

Let Us Teach Guessing

Thirty-five years ago, George Pólya wrote a brief paper with the memorable title "Let Us Teach Guessing." Too few teachers actually do that: most teachers—the overwhelming number—are authoritarian. Teachers set the problems; students solve them. Good students soon learn that the key to school mathematics is to discern the right answer; poor students soon give up.

But Pólya says: let us teach guessing. It is not differentiation that our students need to learn, but the art of guessing. A month spent learning the rules of differentiation reinforces a student's

ability to learn (and live by) the rules. In contrast, time spent making conjectures about derivatives will teach a student something about the art of mathematics and the science of order.

With the aid of a mathematics-speaking computer, students can for the first time learn college mathematics by discovery. This is an opportunity for pedagogy that mathematics educators cannot afford to pass up. Mathematics is, after all, the science of order and pattern, not just a mechanism for grinding out formulas. Students discovering mathematics gain insight into the discovery of pattern, and slowly build confidence in their own ability to understand mathematics.

Formerly, only students of sufficient genius to forge ahead on their own could have the experience of discovery. Now, with computers as an aid, the majority of students can experience the joy of discovery for themselves. Only when the computer is used as an instrument of discovery will it truly aid the learning of mathematics.

With a mathematics-speaking computer, students can now learn college math by discovery.

Metaphors for Mathematics

Two metaphors from science are useful for understanding the relation between computer science and mathematics in education. Cosmologists long debated two theories for the origin of the universe—the Big Bang theory, and the theory of Continuous Creation. Current evidence tilts the cosmology debate in favor of the Big Bang. Unfortunately, this is all too often the public image of mathematics as well, even though in mathematics the evidence favors Continuous Creation.

The impact of computer science on mathematics and of mathematics on computer science is the most powerful evidence available to beginning students that mathematics is not just the product of an original Euclidean big bang, but is continually created in response to challenges both internal and external. Students today, even beginning students, can learn things that were simply not known twenty years ago. We must not only teach new mathematics and new computer science, but we must also teach the fact that this mathematics and computer science is new. That's a very important lesson for laymen to learn.

The other apt metaphor for mathematics comes from the history of the theory of evolution. Prior to Darwin, the educated public believed that forms of life were static, just as the educated public of today assumes that the forms of mathematics are static, laid down by Euclid, Newton, and Einstein. Students learning mathematics from contemporary textbooks are like the pupils of Linnaeus, the great eighteenth-century Swedish botanist: they see a static, pre-Darwinian discipline that is neither growing nor evolving. For most students, learning mathematics is an exercise in classification and memorization, in labelling notations, definitions, theorems, and techniques that are laid out in textbooks like so much flora in a wonderous if somewhat abstract Platonic universe.

Students rarely realize that mathematics continually evolves in response to both internal and external pressures. Notations change; conjectures emerge; theorems are proved; counterexamples are discovered. Indeed, the passion for intellectual order combined with the pressure of new problems—especially those posed by the computer—force researchers to be continually creating new mathematics and archiving old theories.

The practice of computing and the theory of computer science combine to change mathematics in ways that are highly visible and attractive to students. This continual change reveals to students and laymen the living character of mathematics, restoring to the educated public some of what the experts have always known—that mathematics is a living, evolving component of human culture.

Bibliography

Bolter, J. David. *Turing's Man: Western Culture in the Computer Age.* Chapel Hill: University of North Carolina Press, 1984.

Committee on the Undergraduate Program in Mathematics. *Recommendations for a General Mathematical Sciences Program.* Mathematical Association of America, 1980.

Fey, James T., et al., eds. *Computing and Mathematics: The Impact on Secondary School Curricula.* National Council of Teachers of Mathematics, 1984.

Jaffe, Arthur. "Ordering the Universe: The Role of Mathematics." In: *Renewing U.S. Mathematics.* Washington, DC: National Academy Press, 1984.

Koerner, James D., ed. *The New Liberal Arts: An Exchange of Views.* Alfred P. Sloan Foundation, 1981.

Ralston, Anthony. "Computer Science, Mathematics, and the Undergraduate Curricula in Both." *American Mathematical Monthly* **88** (1981): 472–85.

Ralston, Anthony, and Young, Gail S. *The Future of College Mathematics.* New York: Springer-Verlag, 1983.

Shaw, Mary, ed. *The Carnegie-Mellon Curriculum for Undergraduate Computer Science.* New York: Springer-Verlag, 1985.

Steen, Lynn Arthur. "1 + 1 = 0: New Math for a New Age." *Science* **225** (1984): 981.

Wilf, Herbert. "The Disk with the College Education." *American Mathematical Monthly* **89** (1982): 4–8.