Living with a New Mathematical Species

Lynn Arthur Steen

The history of mathematics can be viewed as a counterpoint between the finite and the infinite, between the discrete and the continuous. Although rooted in geometry, Greek mathematics was primarily finite, concrete, and specific. Modern mathematics, in contrast, is infinite, abstract, and general. Aristotle inveighed against the actual infinite, reflecting the Greek cultural distaste for the incomplete form. Centuries later, Leibniz and Newton overcame Aristotelean scruples in proposing methods of calculating with infinitesimals. Now, after three hundred years of Newtonian mathematics, computers are forcing a return to mathematical preferences of the pre-Newtonian age—to the finite, the specific, and the concrete.

This return to our roots is a natural consequence of increasing mathematical maturity. Weierstrass resolved the paradox of infinitesimals by reducing analysis to arithmetic; he showed how to interpret the infinite concepts of calculus in terms of the finite structures of arithmetic. Twentieth century mathematics has been dominated by the Weierstrass synthesis—a working intellectual compromise between the finite limitations of human mental processes and the infinite visions of human imagination.

Today we are forging a new compromise—or in Thomas Kuhn's terms, a new paradigm—binding computers with mathematics. 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* [4] calls computers "embodied mathematics." Computers restore the specific and concrete to the ethereal world of mathematics, yet their very existence depends in crucial ways on the abstract and the theoretical. Although computers would never have been invented without the theoretical support of abstract, continuous Newtonian mathematics, both computer architecture and computer science depend primarily on finite and discrete methodology. Bolter describes the situation more colorfully: "The computer specialist has as little use for irrational numbers as the Pythagoreans had" [4, p. 64].

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. Despite the weight of tradition, mathematics curricula and pedagogy must also change to reflect this new reality.

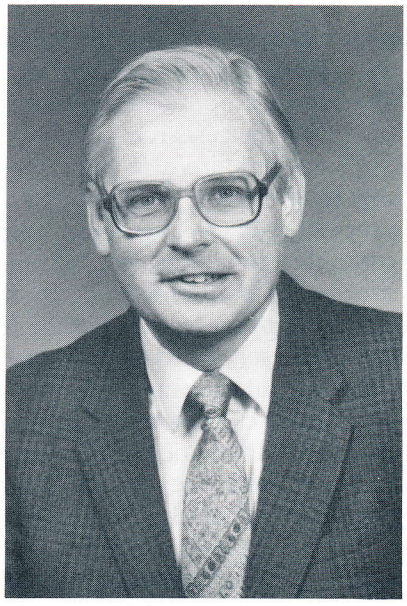
The Ecology of Mathematics

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 ex-

Lynn Arthur Steen



ponentiation, 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-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, 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 Arthur Jaffe has argued so well in his recent essay "Ordering the Universe" [12], 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 change mathematics from an *a priori* to a contingent, fallible subject (see, e.g., [27]), careful analysis reveals that nothing much had really changed. The human practice of mathematics has always been fallible; now it had 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 [8]. 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:

> [Turing] proved that there was no 'miraculous machine' that could solve all mathematical problems, but in the process he had discovered something almost equally miraculous, the idea of a universal machine that could take over the work of any machine. He argued that anything performed by a human computer could be done by a machine. [11, p. 109]

It has been fifty years precisely since Turing developed his scheme of computability [26] 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 25 years for rudimentary machines to demonstrate in a productive way the genius of Turing's idea.

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 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 was enlisted to represent finite automata and recursive functions:

> A quite formalistic approach is now both feasible and desirable, and nowhere is the transition of programming from art to science made more evident. One result of this more formal, disciplined approach . . . is a sharp reduction in the programming effort needed to implement a compiler. [2, p. 423]

Once again, as happened before with physics, mathematics became more efficacious by becoming more abstract.

The Core of the Curriculum

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. The presence of a new and vigorous subject such as computer science produces enormous strains on faculty, curriculum, and resources. As different ecosystems respond in different ways to the presence of a new predator, so different institutions are responding in different ways to the incursion of computer science into the undergraduate 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 [5]. Following two years of calculus and linear algebra, students took core courses in real analysis and abstract algebra (usually two apiece) and selected electives from among such options as topology, differential equations, geometry, complex analysis, number theory, probability, and mathematical statistics. While the faculty expectations and student performance on these courses varied greatly from institution to institution, consensus on a central core was always present.

The subsequent decade was good to mathematics education. The number of bachelor's degrees in the United States rose to about 25,000; the number of Ph.D.s rose gradually from the low hundreds to over 1200. But while core mathematics was experiencing a renaissance, those exploring the frontiers detected evidence of coming change.

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 ("a course introduced over 10 years ago at Harvard by Howard Aiken while director of our computation laboratory"). He also advocated increased use of computer methods in pure mathematics:

> To my mind the use of computers is analogous to the use of logarithm tables, tables of integrals, . . . or carefully drawn figures. 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. Moreover, they give a much more adequate idea of the range to which the ideas expressed are applicable than could be given by a purely deductive general discussion unaccompanied by carefully worked out examples.
>
> Therefore, I believe that [computer-based] courses should be considered as basic courses in *pure* mathematics, to be taken by all students wishing to understand the power (and limitations) of mathematical methods. [3, p. 651]

Rosser emphasized many of the same points, and warned of impending disaster to undergraduate mathematics if their advice went unheeded:

> Unless we revise the calculus course and the differential equations course and probably the linear algebra course . . . so as to embody much use of computers, most of the clientele for these courses will instead be taking computer courses in 1984. . . . If students cannot acquire the necessary computer proficiency and understanding in their mathematics courses, they will have no choice but to take computer courses instead. [21, p. 639]

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.

The symbol of reformation has become discrete mathematics. Several years ago Anthony Ralston argued forcefully the need for change before both the mathematics community [17] and the computer science community [18]. 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 (see, e.g., [19] and [24]), and the Sloan Foundation funded experimental curricula at six institutions to encourage development of discrete-based alternatives to standard freshman calculus.

Five years ago CUPM issued a new report, this one on the Undergraduate Program in Mathematical Sciences [6]. 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. Rather, a department's program must be tailored according to its perception of its role and the needs of its students." The committee did agree that students need to learn to think mathematically and to study some mathematical subject in depth. But they could not agree, for example, that every mathematics major needs to know real analysis, or group theory, or any other topic formerly part of the advanced core of the major.

The niche of mathematics in the university ecosystem has been radically transformed by the presence of computer science in the undergraduate curriculum. As each institution reacts to particular local pressures of staff resources and curriculum tradition, the undergraduate mathematics major has disintegrated into countless local varieties.

Despite the pressure for radical change, the momentum of tradition still permits the strongest mathematics departments to continue the traditional CUPM major for a declining number of students. Reduced enrollment does make it difficult, however, to provide advanced core mathematics courses on a regular basis. In larger institutions, computer science operates as a parallel program, almost always attracting large enrollments, including some of the best and brightest students on campus. It is not uncommon for undergraduate majors in computer science to outnumber mathematics majors by ratios of 20:1 or more.

At smaller institutions a different pattern has emerged. Many such departments have been forced to drop regular offerings of such former core courses as topology, analysis and algebra. 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 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 technological literacy of laymen—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 Stephen White, a program officer with the Alfred P. Sloan Foundation, initiated debate on the proper role of applied mathematics and computer experience in the education of students outside the technical fields. He termed these "the new liberal arts:"

> The ability to cast one's thoughts in a form that makes possible mathematical manipulation and to perform that manipulation, coupled with the fruits of that analysis, are modes of thought. . . . In making use of those modes of thought one may think with enormous new efficiency. But it is thinking itself that is the creative element: thoughtless modelling and thoughtless computation, impressive as they may be, are devoid of real significance. . . . It is precisely as modes of thought that they become essential in higher education, and above all in liberal education [14, p. 6].

Others echoed this call for reform of liberal education. David Saxon, President of the University of California 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" [22].

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 Shake-

speare 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.

As computer science joins mathematics as a basic ingredient in secondary and higher education, liberal education must move beyond computer literacy. As mathematics employs the abstractions of algebra and geometry as tools for problem solving, so courses in computing must incorporate the abstract structures of computer science—algorithms, data structures—in a pragmatic, problem-solving environment. Such computer principles, firmly rooted in mathematics, are a legitimate and important component of the school and college curriculum for all students.

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 exemplar 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 an excellent composite report on the undergraduate computer science curriculum at CMU, surely one of the very best available anywhere. 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. We distinguish here two elements of mathematical thinking that are also crucial to computer science . . . the duel techniques of *abstraction and realization* and of *problem-solving*. [23, p. 55]

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 as well emulate cellular automata or the Ising model for ferromagnetic materials [10]. 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 [1]. An electronic spreadsheet—the archetype of

applied computing—is a structured form for recursive procedures—the fundamental tool of algorithmic mathematics. It is a realization of abstract mathematics, and carries with it much of the power and versatility of mathematics.

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 [25, p. 18].

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. Jim Frey, coordinator of two of the most recent assessments ([7], [9]), 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. [9, p. vii]

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 argues (in [28]) that it is only a matter of time before students will carry with them a device to perform all the algorithms of undergraduate mathematics. Richard Rand, in a survey [20] of applied research based on symbolic algebra agrees: "[Computer algebra] is virtually absent from undergraduate education in the sciences and engineering. . . . however, it is destined for a major role in engineering and applied mathematics. 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 (in [13]) 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 unenlightening. 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.

Following the Rules

Mathematics students, perhaps more than 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'Hôpital's 1699 classic. They present rules of differentiation and integration, with or without proof: linearity, product and quotient rules, chain

rule, substitution, etc. 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 to train students to learn rules. Strong students master them quickly, and yearn for tough problems that extend the rules (e.g., to x^x). Weak students work by rote, carefully adhering to template examples. Students of all types flounder when presented with "word problems" with which to "apply" their skills: "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. They cannot, however, 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 to teach efficiently 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

35 years ago George Pólya wrote a brief paper with the memorable title "Let Us Teach Guessing" [16]. Too few of us 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. It also, almost incidentally, teaches a computational skill of diminishing scientific value. In contrast, time spent making conjectures about derivatives will teach a student something about the art of mathematics and the science of order, in the context of a useful but increasingly unnecessary computational skill.

Imagine a class with access to a good symbolic calculus package. Instead of providing rules for differentiation and exercises to match, the instructor can give motivational lectures replete with physical and geometric interpretation of the derivative. The homework can begin with exploratory questions: ask the computer for the derivative of simple functions. Make conjectures and try them on the machine. After mastering linear functions, try products, then exponentials. Make conjectures; test them out.

The class can discuss their conjectures. Most will be right; a few will not be. Discussion will readily elicit counterexamples, and some informal proofs. With the aid of the 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 for themselves the joy of discovery. Only when the computer is used as an instrument of discovery will it truly aid the learning of mathematics.

Metaphors for Mathematics

Two metaphors from science are useful for understanding the relation between computer science, mathematics, and 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 Contin-

uous 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 20 years ago. We must not only teach new mathematics and new computer science, but we must teach as well 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. Learning mathematics for most students is an exercise in classification and memorization, in labelling notations, definitions, theorems, and techniques that are laid out in textbooks as 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 continually create new mathematics and archive old theories.

Until recently, mathematics evolved so slowly and in such remote frontiers that students in elementary courses never noticed it. The presence of computers in the mathematical ecosystem has changed all that: evolution of theories and notation now takes place rapidly, and in contexts that touch the daily lives of many students. Mathematics itself is changing in response to this intruding species. So must mathematics curriculum and mathematics pedagogy.

Department of Mathematics
St. Olaf College
Northfield, MN 55057

References

1. Arganbright, Dean E. *Mathematical Applications of Electronic Spreadsheets*. McGraw-Hill, 1985.
2. Beckman, Frank S. *Mathematical Foundations of Programming*. The Systems Programming Series, Addison Wesley, 1984.
3. Birkhoff, Garrett. "The Impact of Computers on Undergraduate Mathematics Education in 1984." *American Mathematical Monthly* 79 (1972) 648–657.
4. Bolter, J. David. *Turing's Man: Western Culture in the Computer Age*. University of North Carolina Press, Chapel Hill, 1984.
5. Committee on the Undergraduate Program in Mathematics. *A General Curriculum in Mathematics for Colleges*. Mathematical Association of America, 1965.
6. Committee on the Undergraduate Program in Mathematics. *Recommendations for a General Mathematical Sciences Program*. Mathematical Association of America, 1980.
7. Corbitt, Mary Kay, and Fey, James T. (Eds.). "The Impact of Computing Technology on School Mathematics: Report of an NCTM Conference." National Council of Teachers of Mathematics, 1985.
8. Eckmann, J.-P., Koch, H., and Wittwer, P. "A Computer-assisted Proof of Universality for Area-preserving Maps." *Memoirs of the American Mathematical Society*, Vol 47, No. 289 (Jan. 1984).
9. Fey, James T., *et al.* (Eds.). *Computing and Mathematics: The Impact on Secondary School Curricula*. National Council of Teachers of Mathematics, 1984.
10. Hayes, Brian. "Computer Recreations." *Scientific American* (October 1983) 22–36.
11. Hodges, Andrew. *Alan Turing: The Enigma*. Simon and Schuster, 1983.
12. Jaffe, Arthur. "Ordering the Universe: The Role of Mathematics." In *Renewing U.S. Mathematics*, National Academy Press, Washington, D.C., 1984.
13. Kemeny, John G. "Finite Mathematics—Then and Now." In Ralston, Anthony and Young, Gail S. *The Future of College Mathematics*. Springer-Verlag, 1983, pp. 201–208.
14. Koerner, James D., (Ed.) *The New Liberal Arts: An Exchange of Views*. Alfred P. Sloan Foundation, 1981.
15. Lewis, Harry R. and Papadimitriou, Christos H. *Elements of the Theory of Computation*. Prentice-Hall, 1981.
16. Pólya, George. "Let Us Teach Guessing." *Etudes de Philosophie des Sciences*. Neuchatel: Griffon, 1950, pp. 147–154; reprinted in *George Polya: Collected Papers*. Vol IV, MIT Press, 1984, pp. 504–511.
17. Ralston, Anthony. "Computer Science, Mathematics, and the Undergraduate Curricula in Both." *American Mathematical Monthly* 88 (1981) 472–485.
18. Ralston, Anthony and Shaw, Mary. "Curriculum '78: Is Computer Science Really that Unmathematical?" *Communications of the ACM* 23 (Feb. 1980) 67–70.
19. Ralston, Anthony and Young, Gail S. *The Future of College Mathematics*. Springer-Verlag, 1983.
20. Rand, Richard H. *Computer Algebra in Applied Mathematics: An Introduction to MACSYMA*. Research Notes in Mathematics No. 94, Pitman Publ., 1984.
21. Rosser, J. Barkley. "Mathematics Courses in 1984." *American Mathematical Monthly* 79 (1972) 635–648.
22. Saxon, David S. "Liberal Education in a Technological Age." *Science* 218 (26 Nov 1982) 845.
23. Shaw, Mary (Ed.) *The Carnegie-Mellon Curriculum for Undergraduate Computer Science*. Springer-Verlag, 1984.
24. Steen, Lynn Arthur. 1 + 1 = 0: New Math for a New Age. *Science* 225 (7 Sept. 1984) 981.
25. Travers, Kenneth, *et. al. Second Study of Mathematics: United States Summary Report*. University of Illinois, September 1984.
26. Turing, Alan M. "On Computable Numbers, with an Application to the Entscheidungsproblem." *Proc. London Math. Soc.* 2nd Ser., 42 (1936) 230–265.
27. Tymoczko, Thomas. "The Four Color Problem and its Philosophical Significance." *Journal of Philosophy* 76:2 (1979) 57–85.
28. Wilf, Herbert. "The Disk with the College Education." *American Mathematical Monthly* 89 (1982) 4–8.