8. C. Segre, *Intorno alla storia del principio di correspondenza e dei sistemi di curve*, Bibl. Math. **6** (1982), 33–47 = *Opere*, Vol. 1, Edizioni Cremonese, Rome, 1957, pp. 185–197.

9. H. G. Zeuthen and M. Pieri, *Géométrie enumerative*, Encyclopédie des Sciences Mathématiques, III.4, Gauthier-Villars, Paris, 1915.

STEVEN L. KLEIMAN

BULLETIN (New Series) OF THE
AMERICAN MATHEMATICAL SOCIETY
Volume 12, Number 1, January 1985

0025-5718/85 $1.00 + $.25 per page

Mathematical experiments on the computer, Academic Press, New York, 1982, xvii + 525 pp., $39.50. ISBN 0-1230-1750-5

Mathematics is ancient, the computer is new. Both involve the expenditure of energy to manipulate symbols and thereby create structure. At the very least they complement each other, though they often become corrupting influences. Hilbert hoped that mathematics could be managed by computation; Godel proved that it couldn't. In some quarters limitations on the management of computation, e.g., program verification, by mathematics is taken as a measure of the immaturity of our understanding of computation. Rejecting those august heights of mutual absorption, we find that a pragmatic use of the one by the other is healthy for both.

This book deals with one such use: the computer as a calculator to lend credence to conjectures that could be fragments of theorems or suggestive of theorems. The conjectures have in common a significant computational component. None involves the use of the computer as primarily a manipulator of formuale or logical truths. The author is careful to point out that he is considering experiments that are driven by numerical computation. In a more perfect world an experiment on the computer could invoke both the formula manipulation world exemplified by MACSYMA [**1**] and that recommended by the author. Unfortunately the world of the former speaks LISP [**2**] and the latter speaks APL [**3**] and the two in each other's presence are tongue-tied. Some day....

Conjectures of any kind tend to be plastic and tentative. The programming language and computing environment used in exploration should have the same behavior—programs should be easy to write, test, generalize, and discard. Even more, they should be terse and support a great deal of implicit logical and mathematical structure that would otherwise need to be made explicit by programming. The author's choice of APL is to be applauded. The APL environment, its workspace and library, the terseness of its programs, and the ease with which APL programs can be tested and modified make it the best available language for the purposes Grenander has in mind. Unfortunately there is a viper in this Garden of Eden: Learning to program well in APL is considerably more difficult than in any other language. Even though the mastery pays big dividends, most casual users are unwilling to make the required initial time investment. Other programming languages such as BASIC

and PASCAL are easier to learn, are more widely taught, and are available on more computers. For the purposes described in this book, their use would soon lead one to permanently abandon the computer. Those languages would reduce significantly the favorable balance between mathematics and programming effort that is fundamental to Grenander's use of the computer. Not only is it true, as Grenander points out, that a good experiment has analytic consequences, but the programming effort must not overwhelm the analytic effort. Computers should not turn mathematicians into clerks. In the hands of a good mathematician—programmer, APL is a stiletto, while BASIC and PASCAL are cleavers.

The author devotes about one-fourth of the book to a mathematician's guide to APL. The treatment is prosaic: The elementary properties of APL functions and data are illustrated and supported by exercises. Almost nothing is presented about APL programming styles and techniques. The reader will have to divine his own from an examinaton of a rather extensive collection of APL functions of some mathematical utility that Grenander and his students have collected. These one hundred odd functions will prove invaluable to an experimenter, provided he learns how to express himself in APL and learns how to arrange a computational activity into an organized collection of cooperating functions. Each experiment creates a system to unite a set of independent functions, so that one may creep up on the solution of an incompletely specified problem. Grenander is an excellent programmer, and it is the use of that talent that the reader should devote his attention to.

One of the examples, an experiment in heuristic asymptotes, illustrates the influence of programming (not programming language) on the conduct of an experiment. The author is searching for an asymptotic form that fits given data. The forms are to be constructed from combinations of elementary and arithmetic functions and limited composition. It is natural to consider forms as phrases in a language whose grammar details the rules of generation and similarity. A program can map from property to phrase and then to an APL expression whose computation measures fit. The degree of automation in the interior of the experimental "loop" always requires a balance between the mathematician's insight and the adequacy of a program generated domain of possibilities: The mathematician determines the shifting balance between mathematics and programming that fits each problem somewhere between classical computation and artificial intelligence and gives the experimental apparatus a velocity.

The author is a gifted mathematician who has become a gifted programmer, thereby adding a powerful tool to his collection. This book indicates how the computer can become a powerful adjunct to mathematics research.

The reader should not assume that significant need for numerical computation to test a conjecture is a pre-condition for use of the computer in experiments. The computer can be used in the definition of any set of symbolic structures whose generation is algorithmic. The evaluation strategy—numerical, statistical or symbolic—of relations applied to set members to support an analytic conjecture can usually be algorithmic and, hence, programmed. Unfortunately one may have to abandon APL in favor of LISP when generation

and testing becomes largely symbolic. More need by mathematicians for a mixed APL LISP environment may push computer science to develop a good one. In the world of systems, traffic, not theory, promotes development.

Since a significant part of mathematics education deals with conjecture and proof, this book suggests how the computer could play an important part. For example, the study of calculus, both elementary and advanced, would benefit enormously from the inclusion of an experimental component that goes far beyond the usual elementary numerical analysis applications. Grenander is implying that with skill in programming and use of APL, the experiments can be significant and the programming labor need not dominate the effort required to master either the art or mechanics of mathematics.

References

1. J. Moses, *Algebraic simplification, a guide for the perplexed*, Comm. ACM **14** (1971), 527–537.
2. R. Wilensky, *Lispcraft*, Norton, New York, 1984.
3. L. Gilman and A. Rose, *APL—an interactive approach*, Wiley, 1984.

ALAN J. PERLIS

BULLETIN (New Series) OF THE
AMERICAN MATHEMATICAL SOCIETY
Volume 12, Number 1, January 1985

0025-5718/85 $1 00 + $ 25 per page

Chain conditions in topology, by W. W. Comfort and S. Negrepontis, Cambridge Tracts in Mathematics, Vol. 79, Cambridge University Press, New York, 1982, xiii + 300 pp., $39.50.

Set theory and topology have been bedfellows for a long time. Hausdorff's classic text *Mengenlehre* [**2**], for example, devotes only four chapters to set theory; the remaining six, which comprise three-quarters of the book, deal with point-set topology, especially the theory of metric spaces. Perhaps a better translation of the title would be *The theory of point sets*. A similar approach is found in Kuratowski's book [**3**], except that he devotes even less space to set theory, and he has the decency to entitle the book *Topologie*. And while these books were being composed, Sierpinski was gathering the material, largely topological, which would make up his book [**5**] on the continuous hypothesis.

More recently, after a period during which the two subjects developed separately, there has been a dramatic *rapprochement*, the unifying factors being Cohen's discovery of forcing and the subsequent explosion of work in set theory. Consider, for example, the history of the normal Moore space conjecture, which asserts that every normal Moore space is metrizable. (See M. E. Rudin's monograph [**4**] for an account of all but the most recent parts of this story.) First F. B. Jones showed that if $2^{\aleph_0} < 2^{\aleph_1}$, then every *separable* normal Moore space is metrizable. Then Bing showed that if there is Q-set, i.e., an uncountable set of real numbers every subset of which is a relative F_σ, then there is a nonmetrizable separable normal Moore space, and later Silver deduced from Martin's Axiom (see below), which had recently been shown