SECOND SCHOOL ON ADVANCED TECHNIQUES
IN COMPUTATIONAL PHYSICS
(18 January - 12 February 1988)

SMR.282/ 20

The REDUCE Algebraic Computation System

A.C.HEARN
The RAND Corporation, Santa Monica, USA

(Paper to be published in the Proceedings of the International Workshop on Computing Tools for Research and Development in Sciences and Engineering, held in Capri, Italy, 13-17 April, 1987)

The REDUCE Algebraic Computation System

Anthony C. Hearn

The RAND Corporation
Santa Monica, CA 90406 U.S.A.

REDUCE is an interactive software system designed for general mathematical computations of interest to physicists, mathematicians and engineers. It traces its beginnings to 1963, when I was undertaking research in elementary particle theory at Stanford University. John McCarthy, a professor of computer science at Stanford and the inventor of Lisp, suggested to me that his language was well-suited for automating the hand calculations I was doing, and after a few experiments I became convinced that he was correct. I have been working in the algebraic computation field ever since.

The earliest algebraic computation code I wrote was concerned with the properties of various diagrammatic representations of the physical theories I was studying (1). It quickly became obvious that the techniques were quite general, and in 1968 the first paper describing a general algebra system "REDUCE" was published (2). "REDUCE" is not an acronym although I continue to spell it in capital letters. The name is actually a joke: algebra systems then, as now, often produce very large expressions for many problems, rather than reduce the results to a more manageable form. "REDUCE" seemed to be the right name for such a system.

The first version of REDUCE was quite primitive by today's standards. However, from the beginning it involved a mix of both algorithmic and pattern matching techniques, both of which were necessary for solving the class of problems being considered. It was shortly replaced by REDUCE 2, which first appeared in 1970. The big breakthrough in this release was that the whole system was written in an ALGOL-like dialect, now called Rlisp, rather than in the rather awkward parenthesized notation of Lisp of the original REDUCE. By this time, the system was being distributed to other users, thus marking the beginnings of a user community.

Whereas REDUCE 2 was essentially the work of a single person, REDUCE 3, first distributed in 1983, included several significant packages that were the work of others, in particular those for analytic integration, multivariate factorization, arbitrary precision real arithmetic and equation solving. The number of people now enhancing the system is measured in the dozens, plus the hundreds of people who report problems or suggest improvements.

Since the techniques of computer algebra continue to evolve, an algebra system must also evolve to keep pace with these developments. As a result, we try to release a new version of REDUCE at yearly intervals. A new release contains a combination of new capabilities, improved programming techniques and bug fixes. The present version provides facilities for exact integers, arbitrary precision real numbers, complex numbers and algebraic numbers. It also supports the evaluation, substitution, expansion, simplification, factorization, differentiation and integration of polynomials, rational functions and general algebraic expressions. Built-in matrix algebra provides for the evaluation of determinants and inverses, resultants, eigenvalues and eigenvectors, and the solution of linear equations with algebraic coefficients. Facilities for exterior calculus computations, and for high energy physics calculations are also provided. The system can also output expressions in a FORTRAN, C or Pascal format for direct use in numerical computations.

In addition to these specific capabilities, the program contains a powerful general purpose pattern matching facility to permit the introduction of user-defined simplification rules and side relations, and to perform algebraic computations not provided by the built-in routines. Further control over the simplification process is provided by switches, which control, for example, the expansion of expressions or the cancellation of greatest common divisors. Switches also provide for a variety of output formats, by grouping parts of expressions as required by the user.

From the beginning, REDUCE was designed with a number of definite goals in mind. One such goal was portability. In the early days, the computing requirements were relatively high with respect to the resources available. As a result, I had to be able to use whatever computing equipment was available as effectively as possible. Using Lisp provided a certain level of portability; in other words, any machine that had a Lisp processor could be used. However, as time went on, the underlying Lisp language began to evolve into different dialects. Consequently, the availability of a "Lisp" on a given computer no longer guaranteed that I could use that machine. In order to compensate for this, I limited the REDUCE implementation to a specific subset of Lisp that one could find either directly or by simple mappings in any of the available Lisp implementations. Thus developed Standard Lisp (3,4), a uniform subset of Lisp that could be easily implemented on any computer that already supported a working Lisp system. Initially, we would map the Standard Lisp subset onto the Lisp of the target machine. However, as more programming tools were written in Standard Lisp itself, culminating in a portable Lisp compiler in 1981 (5), we chose instead to force the Lisp on which were running to conform to the Standard Lisp definitions, so we were running Standard Lisp itself rather than some other dialect.

One consequence of this activity has been that when implementors target a new machine for Lisp development, they know that they can run REDUCE if they remain compatible with the Standard Lisp protocols. In particular, they can use the portable compiler for producing high quality efficient code. A look at statistics we have collected for running REDUCE on a variety of different computing systems that use the portable compiler show a strong correlation between the published execution speeds of the machines and the times for running standard tests (6). Invariably, the times for other Lisp implementation models are slower than the Standard Lisp model.

The Lisp standards we adopted have enabled us to implement REDUCE on a wide variety of machine architectures with essentially no changes in the REDUCE sources themselves. Of course each machine requires some system dependent support to allow for differences in such things as input and output and character formats. However, this support has been kept to a minimum. In this manner we have been able to implement the full REDUCE on at least twelve different architectures ranging in power from the IBM PC to the Cray X/MP, a difference in the ratio of REDUCE execution speeds of over five hundred!

Another goal that has been very important in the REDUCE development has been modularity. In order for a system as large and complicated as REDUCE to be maintained and extended, it is necessary that new facilities can be added without requiring changes to the existing code. In particular, a knowledgeable user should be able to add a new application with some assurance that his program can coexist with the existing code. To achieve this goal, many of the facilities in REDUCE 3 are written to depend only on the underlying Standard Lisp subset, and interfaced to the rest of the system through entries in various system tables. In this manner, we were able to add facilities for handling various types of arithmetic such as arbitrary precision real numbers without requiring any major system changes (7).

From the start, REDUCE was designed to be used interactively. This is not unusual in these days of personal computers, but was not common when the system was first written. Many users can solve problems by simply writing expressions, where necessary augmenting the built-in capabilities of the system by straightforward rules defining the particular transformations needed. REDUCE also provides the user with a complete programming language that includes control structures such as FOR and WHILE, block structures, procedure definition, and a variety of algebraic and symbolic data types. The entire REDUCE program and most support packages are written in this language.

REDUCE has been installed on over 1000 main-frame computers world-wide, as well as a growing number of workstations and personal computers. The use of personal machines for algebraic computation should grow tremendously in the years ahead, given their relative low cost and suitability for such calculations (8).

There is a now a well-established base of knowledge about the use of the program in a wide variety of application areas, including quantum electrodynamics and quantum chromodynamics, electrical network analysis, celestial mechanics, fluid mechanics, general relativity, numerical analysis, plasma physics, and a variety of engineering problems such as turbine and ship hull design.

For further information about REDUCE, Gerhard Rayna's recent book (9) is an excellent source.

References

1) Hearn, A.C., "Computation of Algebraic Properties of Elementary Particle Reactions Using a Digital Computer", Communications of the ACM, 9, 573-577, 1966.

2) Hearn, A.C., "REDUCE - A User Oriented Interactive System for Algebraic Simplification", Interactive Systems for Experimental Applied Mathematics, 79-90 (edited by M. Klerer and J. Reinfelds, Academic Press, New York, 1968).

3) Hearn, A.C., "Standard Lisp," SIGPLAN Notices, ACM, New York, 4, No. 9 (Sept. 69). Reprinted in SIGSAM Bulletin, ACM, New York, 13, 1969.

4) Marti, J.B., A.C. Hearn, M.L. Griss, C. Griss, "Standard Lisp Report", SIGPLAN Notices, ACM, New York, 14, 48-68, 1979.

5) Griss, M.L., A.C. Hearn, "A Portable Lisp Compiler", Software Practice and Experience 11, 541-605, 1981.

6) Marti, J.B., A.C. Hearn, "REDUCE as a Lisp Benchmark", SIGSAM Bulletin, ACM, New York, 19, 8-16, 1985.

7) Bradford, R.J., A.C. Hearn, J.A. Padget, E. Schruefer, "Enlarging the REDUCE Domain of Computation", Proc. of the 1986 Symp. on Symbolic and Algebraic Comp., ACM, New York, 1986, 100-106.

8) Hearn, A.C., "The Personal Algebra Machine", Information Processing 80 (Proc. IFIP Congress 80), North-Holland, 621-628, 1980.

9) Rayna, G., "REDUCE Software for Algebraic Computation", Springer-Verlag, New York, 1987.