REDUCE: The First Forty Years*

Anthony C. Hearn

Santa Monica, California, USA

Abstract

Volker Weispfenning and his group in Passau have had a long interest in computer algebra. In this talk, REDUCE, one of the algebra programs they use, will be discussed. The development of this program began over forty years ago. We shall discuss the design decisions that have influenced its long-term survival, and the way in which the program has evolved with time.

1 Introduction

Volker Weispfenning (`www.fmi.uni-passau.de/algebra/staff/w-vita.php3`)began his studies in Mathematics at the University of Heidelberg in 1963. In the same year, I began writing a computer program that would be distributed within a few years as REDUCE. It all began while I was a postdoc in Theoretical Physics at Stanford, working with Feynman diagrams which were becoming increasingly difficult to calculate by hand. I therefore wondered if such calculations could be done by computer.

A new professor in the Stanford Computer Science Department, John McCarthy, gave a talk in the Physics Department proposing his computer language Lisp as the basis for non-numerical calculations in physics. I was out of town, and therefore missed this talk. However, a colleague, knowing of my interest in automating Feynman diagram calculations, suggested I talk to McCarthy about his work. To cut a long story short, McCarthy convinced me that his language would be a suitable basis for such calculations. Part of his efforts to convince me included the offer of free access to a new computer he was acquiring. In those days, getting computer access was not a trivial thing. Computers at that time were expensive with limited access, so such an offer was irresistible! Thus began my work in symbolic computation.

The first publication concerning this work appeared in 1966 [Hea66a]. This paper talked about the specific applications of algebraic techniques to elementary particle physics computations. It quickly became obvious to me, however, that the techniques I was developing were quite general, and in 1968 the first paper describing a general algebra system REDUCE was published [Hea68]. "REDUCE" is not an acronym, although I continue to spell it in cap-

*Invited paper presented at the A3L Conference in Honor of the 60th Birthday of Volker Weispfenning, April 2005.

ital letters[†]. Its name was actually intended as a joke; algebra systems then as now tended to produce very large outputs for many problems, rather than reduce the results to a more manageable form. "REDUCE" seemed to be the right name for such a system.

REDUCE 2 first appeared in 1970. The big change in this release was that the whole system was written in an Algol-like dialect, RLISP, rather than in the rather awkward parenthesized notation of Lisp used for 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 new packages that were the work of others, in particular those for analytic integration, multivariate factorization, arbitrary precision real arithmetic and equation solving. Over a hundred people have now been involved with enhancing the system, plus many more who take the time to report problems or suggest improvements. My heartfelt thanks go to all these people. A list of the contributors may be found on the REDUCE web site, `reduce-algebra.com`.

2 Design Goals

From the beginning, REDUCE was designed with a number of goals in mind, nearly all of which have contributed to its long-term use. One such design goal is portability. In the early days, the computing requirements were relatively high with respect to the resources available. As a result, I had to be sure that I could use whatever computing equipment was available to me. Using Lisp provided a certain level of portability, since most available machines supported it.

As time went on, the underlying Lisp language itself began to evolve into different dialects. Consequently, the availability of a "Lisp" on a given computer no longer guaranteed that I could easily 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 of any of the available Lisp implementations. Thus developed Standard Lisp [Hea66b, MHGG79], 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 [GH79], we chose instead to force the Lisp which we were using to conform to the Standard Lisp definitions. In other words, we were running in Standard Lisp itself rather than some other dialect. Complementing this activity was the development of specific Standard Lisp implementations [GBM82, Nor93]. A description of one of these systems, CSL, may be found in Arthur Norman's paper at this meeting.

One consequence of this activity has been that when implementors target a new machine

[†] In the 1960's, input devices only had capital letters, hence nearly all programs those days had upper-case names.

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 [MH85]. The range of these machines has been quite extreme. In 1985, for example, a PSL implementation on a Cray X-MP, the worlds's most powerful machine at the time, set the standard for timings. At the other end of the spectrum, Arthur Norman recently reported the porting of CSL and REDUCE to a Linksys router costing under $US100, and running the REDUCE tests at about 10% of the speed of the X-MP.

The Lisp standards we adopted have enabled us to implement REDUCE on a wide variety of machine architectures with essentially no changes to the REDUCE sources themselves. Of course, a new machine sometimes requires some system-dependent support to allow for differences in such things as input and output, and character formats. However, coding this support has been kept to a minimum.

Another important REDUCE development goal has been modularity. In order for a system as large and complicated as REDUCE to be maintained and extended, it is desirable 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 or her program can coexist happily 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.

One of the most important attributes of REDUCE is its worldwide acceptance as a useful problem solving tool. As a result, there is a well-established base of knowledge about the use of the program in a wide variety of application areas. These application areas include 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. This maturity provides the user with some assurance about its reliability and ease of use.

3 REDUCE Today

Over forty years after the project began, REDUCE remains in worldwide use, and groups are still developing new code for it. At the present time, REDUCE includes approximately fifty contributed packages. Several of these are still under active development, including some described at `www.redlog.eu` developed by members of Volker Weispfenning's group.

A commitment to release updated and expanded versions of the system was made from the beginning, since I realized that the techniques of computer algebra would continue to

develop and improve, thus requiring an algebra system to change to keep pace with these developments. The complete source code was also distributed to encourage users' understanding of the program. In the early days, new versions of the program were released at fairly regular intervals, often yearly. However, as things matured and the code became more stable, often the only reason for an update was to correct bugs that had been discovered, or to include improved code in a particular package. To obviate the need for a complete new version for these purposes, we began a few years ago to develop a "patches" mechanism, which enables users to download a file from the Internet containing corrections to their version. In that way, a base version could last several years, with the patches file giving users access to improved code on a regular basis. The most recent version of CSL-based REDUCE, which uses system-independent pseudocode for loading the various packages, also allows users to update to the latest version of these patches by a simple menu-driven command.

However, there is a small group of developers, numbering about twenty, but widely distributed geographically, who require more extensive updates to their code than the patch mechanism could provide (e.g., a new package, or a complete reorganization of an existing package). This includes members of Volker Weispfenning's group. To meet this need, Arthur Norman and I developed a robust Web-based mechanism [NH99] that enabled members of the development group to keep their copies of REDUCE in synch. This represents one of several ways in which REDUCE is taking advantage of the possibilities offered by the World Wide Web. Others include online access to demonstration versions, and various related projects. A description of these may also be found at `reduce-algebra.com`.

4 Future Developments

Working with REDUCE over so many years has generally been a rewarding experience for me. However, there have been a few disappointments. One in particular concerns some ideas that are intended to form the basis of a future version of REDUCE (REDUCE 4). In the early 90's, Eberhard Schrüfer drew my attention to some work on order-sorted logic that he thought was a better approach to object-oriented algebraic manipulation than that provided by the existing methods at that time. He convinced me of this, and together we produced a system that supported this method. We presented the ideas on several occasions, and published a paper [HS93] describing them. However, we never had any positive feedback from anyone, and so further development has proceeded slowly. I am hopeful that this version will eventually be distributed, but it needs a lot more work before that happens. If anyone would like to collaborate on this project, I would be happy to discuss the ideas with them and give them access to the existing code.

5 Conclusions

Forty years is a long time for a computer program to survive. During that time it has been of considerable use to a large number of people throughout the world. In this paper I have tried to highlight some of the reasons why the program has lasted so long. Its continuing development will depend to a large extent on the dedicated work of groups like those of Volker Weispfenning, who are still extending the capabilities of the program, thus making it even more useful to its user community.

References

[GBM82] Martin L Griss, Eric Benson, and Gerald Maguire. PSL: A portable lisp system. In *ACM Symposium on Lisp and Functional Programming*, 1982.

[GH79] Martin L. Griss and Anthony C. Hearn. Portable LISP compiler. *Software - Practice and Experience*, 11:541–605, 1979.

[Hea66a] A. C. Hearn. Computation of algebraic properties of elementary particle reactions using a digital computer. *Comm. ACM*, 9(8):573–577, 1966.

[Hea66b] A. C. Hearn. Standard Lisp. *SIGPLAN Notices*, 4(9), 1966.

[Hea68] Anthony C. Hearn. REDUCE: A user-oriented interactive system for algebraic simplification. In M. Klerer and J. Reinfelds, editors, *Interactive Systems for Experimental Applied Mathematics*, pages 79–90, New York, 1968. Academic Press.

[HS93] Anthony C. Hearn and Eberhard Schrüfer. An order-sorted approach to algebraic computation. In *Proc. DISCO '93, Lecture Notes on Comp. Science*, volume 722, pages 134–144. Springer-Verlag, 1993.

[MH85] Jed B. Marti and Anthony C. Hearn. REDUCE as a LISP benchmark. *SIGSAM Bulletin*, 19(3):8–16, August 1985.

[MHGG79] J. B. Marti, A. C. Hearn, M. L. Griss, and C. Griss. Standard Lisp Report. *Sigplan Notices, ACM*, 14(10):48–68, 1979.

[NH99] Arthur C. Norman and Anthony C. Hearn. Synchronization of distributed development software. In *Proceedings of IMAC Workshop, ISAAC '99, Vancouver, Canada*, June 1999.

[Nor93] A. C. Norman. Compact delivery support for REDUCE. In *Proc. DISCO '93, Lecture Notes on Comp. Science*, volume 722, pages 331–340. Springer-Verlag, 1993.

Anthony C. (Tony) Hearn is an adjunct staff member at RAND in Santa Monica, California and at the IDA Center for Computing Sciences in Bowie, Maryland. He received his undergraduate education at the University of Adelaide in Australia before obtaining a Ph.D. degree in Theoretical Physics from Cambridge University in 1962. From 1962 until 1964 he was a Research Associate in Physics at Stanford University, and returned there as an Assistant Professor in 1965 after a year at the Rutherford Laboratory in England. In 1969 he joined the University of Utah as an Associate Professor in Physics, and was made Professor in 1971. From 1973 until 1980 he was Professor and Chairman of the Department of Computer Science. He joined RAND in July, 1980 as Head of the Information Sciences Department, a position he held until August 1984. He was a Resident Scholar at RAND from 1990 until 1996.

hearn@rand.org www.rand.org/personal/hearn