60+ years of Applications: a perspective from Reduce

Arthur Norman
Trinity College
Cambridge, UK
acn1@cam.ac.uk

June 2026

This meeting of ACA notes that it represents 30 years of consideration of the Applications of Computer Algebra. Over that time most of the papers will have concentrated on a presentation how some rather specific application has been addressed using computational tools, and while often the progress described will involve devising new or enhancing existing algorithms, the impact of this on Computer Algebra as a whole will not have been considered. This paper reviews the development and growth on one particular algebra system as prompted by the applications made of it. Very often a software base needs time to stabilize (and the underlying computers have needed to become more powerful) before application can become routine. So here I consider the case of Reduce which as a system has a history spanning 60 years – i.e. twice the lifetime of ACA – and I consider how its development over that extended time period has been driven by a wide range of applications, with that activity continuing up to today.

The lessons that I believe can be drawn from this longitudinal study are:

1. In the early years algebra systems provided fairly basic capabilities, but they found that a very broad range of applications were still within their scope because the calculations involved where huge in scale rather than especially technically challenging per se. This is still often the situation today;

2. Successions of users with the particular scientific problems they were working on have set the agenda for system builders and algorithm designers by identifying particular aspects of symbolic computation that those early systems either did not support at all or where performance was a particular concern;

3. Situations where those who had problems to solve have codified and packaged their work and it has been possible to merge that expertise back into the central system have been important over and over again. Anybody who today sorts out an improved way to make progress in some application domain should be encouraged to contribute what they have done for the benefit of others who follow on;

4. Successful transfer from users as above can benefit from an open system where individuals can observe, access and where necessary modify everything, where the challenge of learning how to extend the system is not too severe, where license terms do not intrude and where system portability means that code developed in one arena will readily migrate to all others;

5. The communication channels from users to developers and maintainers may be almost as important as many fine details of a system when it comes to getting support for a project. This can mean that migration between the three communities should be encouraged;

6. The particular system – REDUCE – discussed here at over 60 years old and still under active development as well as use is among the oldest software systems with such a long history. There are significant parts of the core where the almost original code is still in use. Such a long life is surely a symptom of it having got some things right, and so everybody concerned about how long their own legacy of calculated results and contributed code will last might reasonably want to consider how the close interaction between users with challenging applications and those concerned with the central structure of the system has developed;

7. Anybody who builds their work on the basis of computer algebra might consider whether spending money on a commercial system will guarantee the longer term survival and support of that system. At the same time they can consider the investment of time (sometimes their own, sometimes graduate students or other juniors) in discovering how to build and maintain an open source system. It may sometimes not be clear which will be the better long term policy.

Perhaps the major point that I hope this paper will bring to readers' attention is that Computer Algebra systems do not need to be thought of as either magic or as black boxes, and that over the years a great deal of their development has been driven by those with an applications focus, so ACA can continue to set priorities for the next enhancements to be made, and it potentially provides a pool of expertise that could be developed to gradually take over system support and to provide guidance to future generations.