Restoring the Link to Computer Science

Jacques Calmet
University of Karlsruhe
Postfach 6980
76128 Karlsruhe, Germany
calmet@ira.uka.de

September 28, 2000

Abstract

This sketchy note tries to answer the question put forward "What is your view of the future of Computer Algebra". Part of the answer is to explore better the links with Computer Science.

We stress the fact that a question raised by John Mc Carthy in 1963 on the interaction between Algebra and Logic can set partly an agenda for future research.

To foresee the future of a scientific field is a hazardous game. Trend identification depends in particular on the present state of the art, on the past history of the field, on the research interests of the assessor and on the available technology.

1 A brief history

Computer Algebra (CA) started as a discipline of Computer Science [1]. The first conferences were divided into three sections: systems, algorithms and applications. These sections had almost equal weight. This is best illustrated by the Los Angeles conference in 1971. The application section was very important since it was a way to validate the systems and algorithms being designed. At that time, Mathematics was part of the agenda set up by the founder's of Artificial Intelligence in the late 1950's [2]. Along the years, the application component became weaker. Today, authors of papers on systems have difficulty to get them accepted at ISSAC. Indeed, algebraic algorithms dominate the field now. This may look surprising since the outside world knows CA best from the several successful systems such as Maple or Mathematica. In fact, research on algebraic algorithms has been both successful and flourishing along the years.

The thesis in this short note is that the applications have now found their optimal setting, that research on algebraic algorithms faces some challenging problems and that the decline of the "system" part is due to severed links to Computer Science and needs to be restored.

2 Applications

The founding of the IMACS-ACA series of conferences has been a landmark for presenting applications of CA. Other informal conferences such as the Conference Day in North America or the Rhine Workshop in Europe also enable contacts with applications.

Another important feature is that application domains have drifted away from CA when successful. An illustrative example is Particle Physics, a field that helped to establish CA. Today physicists use routinely CA systems. But, it will probably be FORM. Also, they have their own conferences and journals and no longer show up at formal CA conferences. We may expect and wish that other topical or regional conferences would be founded.

A remaining challenge is how to collaborate better with new application domains. A typical one is Business and particularly Econometrics. Specialized library packages or books are available but there is still a strong demand from this field to establish links. A possible answer is to propose specialized lectures at university level and to submit common research proposals to relevant funding agencies. A safe guess is that more specialized packages will be designed for specific applications.

3 Algebraic Algorithms

Nowadays CA is almost synonymous of algebraic algorithms. Any conference proceedings or table of contents of any topical journal illustrate this point. It is safe to assume that research in this domain will remain very active and of high quality.

A first remark is that the concept of algorithm has many definitions. In CS it usually means a computational procedure that is finite, definite and always terminates. In CA we aim at designing constructive methods that solve all occurrences of a given problem. Symbolic integration is such an example. In addition we require the algorithms to be efficient. The latter remark leads to transform, usually through homomorphic transforms, a problem over a given domain to a problem over a domain where more efficient solutions can be designed.

In general terms, three approaches may be investigated: to improve the efficiency of existing algorithms, to discover new constructive algorithms and to rely on heuristic methods.

A thorough complexity analysis of existing algorithms can discover ways to design more efficient versions. This trend has already started in the TERA project of J. Heintz and co-workers where a complexity analysis leads to find more efficient algorithms for Gröbner bases computation and related problems.

The second approach is mathematically challenging. Two problems provide good examples. A first one is to find close-form Liouvillian solutions of linear homogeneous ODE's. Singer proposed many years ago a general algorithm. Ulmer improved it in 1991 and proved that it is optimal. The task was then, and still

is, to apply it at each order. The solution for order 2 was found independently by Kovacic. To establish it from order 3 on is still both a mathematical and a programming challenge. Part of the challenge is to compute representations of Galois groups. It is worth noticing that we cannot rely on known examples since they do not exist in most cases.

A second example, even more difficult, is to solve partial differential equations.

What is known is to duplicate available methods to solve special classes of problems. We do not know any mathematical framework suitable to design constructive algorithms. On the mathematical side, many of the related problems are open for over a century. A track under investigation is the so-called formal theory of differential equations [3]. A remote possibility is to go a step beyond in the level of abstraction and to look at the theory of algebraic fields to search for an adequate framework.

To rely on heuristic methods is a way to tackle quickly new classes of problems. Users will design many such methods. Candidate problems are those in analysis which cannot be formulated algebraically. Whether they will be integrated in the library of existing systems is difficult to tell. A first conclusion is that the first two approaches point toward the use of very sophisticated mathematical methods. From the point of view of the sociology of sciences this means that we need to collaborate with very high level mathematicians and that it is unlikely that computer science students will ever have the required background to work on such problems. A side remark is that heuristic methods do not mean using AI techniques. Unless one is not aware of what AI is today.

4 Systems

One of the main features of CA is the enormous success story of systems like Mathematica or Maple. By comparison, no user can buy a theorem prover to "play" with it. They simply do not exist as products. Users can download them free of charge but they are very hard and non-intuitive to use. It is well known that successful software products are hardly affected when the software technology evolves. Successive versions provide improvements but the basic architecture remains mainly unaffected. It is again a safe guess that CA systems will verify this rule.

The claim here is that modern methodologies in Computer Science enable to "do" more Mathematics on computers than only CA. Already in 1963, John Mc Carthy proposed the idea that the integration of Algebra and Logic would have the same effect on Science as the integration of Analysis and Physics had in the 20th century.

A key remark toward such a goal is that Mathematics provides a very demanding and non-trivial model to design CS methods. For instance, it is known that there is no unique typing system for Mathematics. CA experts are well aware of the theorem of Richardson and thus do not need a further proof.

But, in CS a system is built upon a model and must be semantically sound

and algebraically specified. CA systems are not semantically sound and not specified. We enumerate a list of problems, which could reset a link between CA and CS. In our opinion this link has become very thin as of today.

The approach of any computer scientist with training in artificial intelligence would be to consider that CA is a knowledge representation problem. A possible model is to define a piece of mathematical knowledge as an operator, its definition domain (its type) and its properties (the specification). The latter is not really useful from a mathe matical point of view but necessary from a CS point of view as a specification tool. This leads to the idea of representing algorithms as schemata's, a concept of AI. Then, this provides a way to define the concept of the context (another import from AI) of a computation. This enables to design semantically sound systems. Despite a few attempts to base typing systems in CA on category theory, the state of the art is still far away from the state of the art in programming language. When a human does Mathematics, he/she usually combines doing symbolic or numerical calculus, proving theorems, consulting books, drawing graphs among other tasks. The most challenging task is to couple CA and provers. Several approaches have been selected [4]. This is a long-term effort that ought to attract the interest of CA. There is a strong connection to Open Math, which aims at integrating CA systems.

As mentioned in [4] this coupling leads to an approach to the specification of computational systems. Then, this is a possible basis to design a mathematical software bus. The task of a software bus is to integrate heteregeneous software systems through communication protocols. An open problem at that time is to specify formally the integration of numerical system. A possible track is to look into interval arithmetic.

This is a short list of problems that are already under investigation. References are found, except for Open Math, in the web page of the author. This list is by no means exhaustive. For instance, it looks promising to investigate techniques used in compiler construction when working on the coupling with numerical computation.

5 Teaching Mathematics

One of the main applications of CA systems is their widespread use in classroom to teach Mathematics. They are very useful to teach calculus for instance. The coupling to theorem provers could provide a tool to teach how to prove theorems. The basic idea is that some systems generate proof plans. They are too long and complex when using stand-alone provers. The benefit of CA systems is to master the growth of such proof plans. One may guess that such an approach will be investigated. Tutoring systems belong to knowledge representation. When the approach outlined in the previous section is adopted, it becomes possible to design intelligent tutoring systems.

6 The impact of technology

At the origin of the success of CA is the availability of personal workstations in the late 1980's. Today, many ambitious projects are becoming feasible since memory is plentiful and cheap. Today, as everyone knows, the new technology is not in hardware but in communication: the Web. This has already an impact on CA and will have much more in the future. What is already underway is an attempt at publishable Mathematics (Open Math). This implies an attempt at standardization. Previous ones have failed in CA. This one ought to succeed. A side remark is that there is a strong need to define a common dictionary of words being used. Many words used in CA and originating in CS often diverge from their original meaning. Another consequence of the existence of the Web is that we may go in the direction of distributed computing. Using terms of CS this means that we will have to query heterogeneous information sources, retrieve information and possibly do data mining. Using (subsets of) Mathematics as a model is then a source of challenging research problems.

This is a personal view of the future of CA. It can be summarized as follows: a need for high level Mathematics on one side and the wish to use Mathematics as a model for designing CS systems on the other side.

References

[1] Michael J. Wester (ed.) *Computer Algebra Systems: A Practical Guide,* John Wiley & Sons, 1999.

[2] J. Calmet and J.A. Campbell. *Perspective on Symbolic Mathematical Computing and Artificial Intelligence.* Annals of Mathematics and Artificial Intelligence. Vol. 19/3-4, pp. 261-277, 1997.

[3] J.-F. Pommaret. *Systems of partial differential equations and Lie pseudogroups.* Gordon & Breach, London 1978.

[4] J. Calmet, invited talk, this conference.