A Systems Perspective on A3L

Heinz Kredel

University of Mannheim, IT-Center, 68131 Mannheim, Germany

Abstract

In this paper we summarize some aspects of the development of computer algebra systems from the last 25 years. We focus on Aldes/SAC-2, MAS and some new developments in Java. It will turn out that computer algebra can more and more use standard software developed in computer science to reach its goals. In this systems theories of Volker Weispfenning have been implemented to varying degrees.

1 Introduction

It was around 1986 when I came to work with Volker Weispfenning. At that time (with some experience in Gröbner Bases) I was pretty much sure that the dimension of a polynomial ideal could be determined by inspection of the head terms of the polynomials in the Gröbner base but was not able to give a rigorous proof. Together we succeeded to find such a proof [KW88].

In the following however, I will not focus on the mathematical work with Volker, but on the perspective of constructing software for the representation and computation of Algebraic Algorithms and Logic. First I will consider the time when I entered Computer Algebra using Aldes/SAC-2, my time in Passau using Modula-2 and the last years using Java. Finally we will draw some conclusions. We will not consider software development style in the days of ubiquitous Internet and Open Source related topics.

2 Aldes/SAC-2

Besides the implementation of the algorithm to compute the dimension of a polynomial ideal via the head terms in its Göbner Base, a major task in the end of these days was the implementation of Volkers theory of Comprehensive Gröbner Bases in DIP by Elke Schönfeld.

The Distributive Polynomial System (DIP) [BGK86] was implemented in Aldes/SAC-2 [CL82], which consisted of an translator for the Algebraic Description Language (Aldes) [Loo76] to standard FORTRAN together with a run time system for list processing with automatic garbage collection. The origin of this system was SAC-1, a pure FORTRAN implementation with a reference count garbage collecting list processing system. Both systems aimed at an implementation of G. Collins cylindrical algebraic decomposition theory (a somewhat practical method for the real quantifier elimination problem) and provided its users with a comprehensive library of fast and reliable algebraic algorithms. On top of these we had implemented one of the first Buchberger algorithms which were not restricted by shortcomings of the used implementation language or static bounds. The system was also used for the implementation of zero-dimensional ideal decompositions and real root computations.

With the advent of the Pascal programming language and the first personal computers it became apparent that one could imagine to implement such a system in a modern interactive development environment like Turbo-Pascal was at this time. We tried on several Pascal compilers but were not able to implement a suitable list processing system. Surprisingly when trying with Modula-2 all obstacles could step by step be overcome.

3 Modula-2

With Modula-2 it was possible to implement run time support for a list processing system with automatic garbage collection as needed. (At the same time Boehm implemented a garbage collector in an uncooperative environment in C.) By reworking the Aldes to FORTRAN translator we implemented a bootstrapping translator to Modula-2 within the Aldes/SAC-2 system. In the following years all of the existing Aldes algorithms (with the exception of one spaghetti code program) were transformed to Modula-2 and built up the Modula Algebra System (MAS) [KP03]. Using Modula-2s procedure parameters an interpreter for the algebraic libraries was developed and an interpreted language similar to Modula-2 was designed. (This concept was also in the origin of the still successful Python programming language.) This interpreter was further used to experiment with language extensions seen in algebraic specification languages (ASL), term rewriting systems, Prolog like resolution calculus and interfacing to numerical (Modula-2) libraries.

During his time many students implemented Volkers theories on real root counting, real quantifier elimination, comprehensive and universal Gröbner bases and solvable polynomial rings as part of their Diploma Thesis. You will hear of all of those theories and implementations in this conference.

With the advent of modern micro-processors, introducing dramatic speed differences between cache and main memory, it became apparent that the MAS list processing is no more suitable for fast performing implementations. During long running computations, the list elements of algebraic data structures (including integers) tend to be scattered randomly throughout main memory, thus leading to cache misses and CPU stalls at every tiny step. Other CA systems facing this problem have been able to replace the implementation of (arbitrary precision) integers with better memory hierarchy aware libraries like Gnu-MP. However there is no transparent way of replacing integer arithmetic in MAS with other libraries due to the ingenious and elegant way G. Collins represented integers in the Aldes/SAC-2 libraries: small integers ($< 2^{29} =$ `beta`) are represented as 32-bit integers of the hosting programming language and when integers grow larger they are transformed to lists. Thus all code dealing with arithmetic is full of case distinctions of the form '`IF i < beta THEN`' which would break if a new data structure for integers would be introduced. Another problem which makes replacement of modules hard is the representation of the *zero* element of nearly all algebraic data structures by the programming language integer '`0`'. On the bottom line there seemed to be no automizable way of replacing basic implementation modules and we were stuck.

After leaving Passau I had the opportunity of working with parallel computers (KSR systems with up to 128 processors on virtual shared memory and others) at Mannheim University. In this time MAS was ported and implemented on this parallel computer starting with a parallel garbage collector for the list processing subsystem. Based on POSIX threads we designed a parallel version of Buchberger's algorithm along with a pipelined version of the polynomial reduction algorithm. However by the very dynamic computational load occurring in Gröbner base calculations it was not possible to obtain reliable speedup on many processors. Due to the tight integration with the KSR system this version of MAS was not portable and was therefore not released. So we ended with a bunch of interesting new problems: respecting the memory hierarchy, load balancing and task granularity of algebraic algorithms and the requirement of some portable way of parallel software development.

4 Java

Since the development the languages of N. Wirth, Modula-2 and Oberon was not as expected, it was clear that MAS had no future in this respect. Other colleagues favored C style languages for the implementation of algebraic software like H. Hong with SACLIB [HNSS93] and W. Küchlin with PARSAC [Küc90]. But they seem to suffer also from the list processing problems mentioned above. Systems using C++, like LiDIA from T. Papanikolaou, or like Singular of H. Schönemann, have incorporated Gnu-MP, but seem to have other problems with the C++ language or the memory management. Some of the colleagues also turned to commercial systems like Maple, Mathematica and Reduce and reimplemented portions of their code in them.

In my search for portable ways of parallel software development I became exposed to the Java programming language. Together with A. Yoshida we drove so far into this subject, that we could publish a book on parallel programming with Java [KY02]. By the time we got confident in the performance of Java implementations and object oriented software development also seemed to be a must for any new projects. So in 2000 a Modula-2 to Java translator came in handy and a port of MAS to Java was started. The first version, still using SAC-2 style list processing directly ported to Java, was about 5-8 times slower on Trinks6 Gröbner base computation than the MAS implementation. But comparing the used integer arithmetic with Java's `BigInteger` class showed an improvement by a factor of 10-15 for Java. This proved that all list processing code has to be abandoned and native Java data structures should be used where ever possible. Polynomials were now reimplemented using `java.util.TreeMap` thus bringing the implementation closer to the theory of polynomials, being a map from a monoid to a coefficient ring. The resulting version is now a factor of 8-9 better on Trinks6 Gröbner base than the MAS implementation on the same machine.

Besides several refactorings to make use of more and more object oriented principles, a shared memory and a distributed memory parallel version for the computation of Gröbner bases have been implemented. The parallel GB algorithm (doing polynomial reductions in parallel) uses a critical pair scheduler as work-queue and the scalability is perfect up to 8 CPUs on shared memory, provided the JVM uses the parallel Garbage Collector and aggressive memory management. The distributed GB algorithm uses the same (central) critical pair scheduler and a distributed hash table for the polynomials in the ideal base with central index managing. Communication of polynomials is easily done using Java's object serialization capabilities. The software will be available at `krum.rz.uni-mannheim.de/jas`. Underway is an implementation of solvable polynomial rings and the use of generics coming in with JDK 1.5.

5 Conclusions

Not all mathematically ingenious solutions like the small integer case can persist in software development. A growing part of software need no more be developed specially for CA systems but can be taken from libraries developed elsewhere by computer science (e.g. STL for C++ or `java.util`). Required programming language features like dynamic memory management with garbage collection, object orientation (including modularization) together with generic data types as well as concurrent and distributed programming are now included in languages like Java or C#. In the beginning of CA systems development only a small part was taken from computer science (namely FORTRAN). Then a bigger part (Modula-2, C++) was employed and today about the half part (Java) can be used from the work of software engineers. The conclusion is to go and use the improvements of computer science and systems engineering for implementation of A3L algorithms. But don't forget to observe and adapt to hardware developments: memory hierarchy, multi-core CPUs and distributed systems.

Acknowledgments

First I heartily thank Volker for the opportunity to work with him and in his group at Passau. Second I thank all colleagues from Passau, especially Th. Becker and M. Pesch, and the computer algebra community for fruitful discussions and their support. I apologize for not mentioning all people and missing references to related work due to the limited space (most references can be found in the Computer Algebra Handbook [GKW03]).

References

[BGK86] W. Böge, R. Gebauer, and H. Kredel. Some examples for solving systems of algebraic equations by calculating gröbner bases. *J. Symb. Comp.*, **2**/1:83–98, 1986.

[CL82] G. E. Collins and R. G. Loos. ALDES and SAC-2 now available. *ACM SIGSAM Bull.*, 12(2):19, 1982.

[GKW03] J. Grabmaier, E. Kaltofen, and V. Weispfenning, editors. *Computer Algebra Handbook*. Springer, 2003.

[HNSS93] H. Hong, A. Neubacher, W. Schreiner, and V. Stahl. The design of the SACLIB / PACLIB kernels. In *DISCO'93, LNCS 722*. Springer, 1993.

[KP03] H. Kredel and M. Pesch. *MAS: The Modula-2 Algebra System*, pages 421–428. in Computer Algebra Handbook, Springer, 2003.

[Küc90] W. Küchlin. PARSAC-2: a parallel SAC-2 based on threads. In *AAECC'90, LNCS 508*. Springer, 1990.

[KW88] H. Kredel and V. Weispfenning. Computing dimension and independend set for polynomial ideals. *J. Symb. Comp.*, **6**/2,3:231–247, 1988.

[KY02] H. Kredel and A. Yoshida. *Thread- und Netzwerk-Programmierung mit Java*. dpunkt, 2nd edition, 2002.

[Loo76] R. Loos. The algorithm description language ALDES (Report). *ACM SIGSAM Bull.*, 10(1):15–39, 1976.

Heinz Kredel got his diploma in Mathematics from University of Heidelberg and received his doctorate in Mathematics and Computer Science from the University of Passau. At present he is leading the working group 'Zentrale Systeme' together with W. Aufsattler at the IT Center of the University of Mannheim. He is working for more than twenty years in the areas of computer system administration and computer algebra with special interests in parallel computation and software development.

`kredel@rz.uni-mannheim.de` `krum.rz.uni-mannheim.de/kredel`