Parallel Computer Algebra Software as a Web Component

Andreas Weber* Wolfgang Küchlin Bernhard Eggers
Volker Simonis
Wilhelm-Schickard-Institut für Informatik
Universität Tübingen, 72076 Tübingen, Germany
E-mail: {weber,kuechlin}@informatik.uni-tuebingen.de
WWW: http://www-sr.informatik.uni-tuebingen.de

Abstract

In the field of computer algebra, joining separate computer algebra systems with individual strengths into problem solving environments is an important recent research direction. In this paper we describe how we wrapped a specialized computer algebra system, which uses shared-memory parallelism to speed up Gröbner basis computations, into a Web component that can be used by other systems. By this component also a new algorithmic development in Gröbner basis computations—the use of the *fractal Gröbner walk*—was made available to various general purpose computer algebra systems in the form of a single optimized implementation. Previously, all of the important algorithmic improvements to an algorithm had to be reimplemented in the general purpose systems . Our implementation allows access to the Gröbner basis computations via "intelligent" graphical user interfaces and via scripting from existing general purpose algebra systems.

1 Introduction

In the field of computer algebra, joining separate systems and algorithmic software components with individual strengths into problem solving environments is an important recent research direction. Currently, the existing systems can be subdivided roughly into two categories.

General purpose systems such as Axiom, Macsyma, Magma, Maple, Mathematica, or Reduce come with nice user interfaces and sophisticated graphical capabilities and contain a broad range of computer algebra algorithms. However, because of the size of the systems and the broad range they cover, they cannot be always up to date algorithmically. Also the software architecture of such a general purpose system—consisting of a kernel and an interpreted language in which many algorithms are written—yields computational overheads that can be avoided by systems optimized for a certain class of algebra problems. Also these systems are in general not suited to make use of special hardware, such as shared-memory multiprocessors.

Specialized systems are usually developed by research groups that concentrate on algorithm research in one special area of computer algebra. These systems usually contain the latest algorithms of these areas and their implementations are very often tuned up considerably. Several of these systems are developed for high-performance computing architectures such as parallel machines. However, these systems usually have a limited scope and crude user interfaces.

* Supported by *Deutsche Forschungsgemeinschaft*.

Connecting specialized systems and general purpose systems are of a major interest for different reasons. First, instead of reimplementing a new algorithm in all of the general purpose systems, a single optimized implementation could be used by them, e. g. on a parallel machine, if the specialized system in which the algorithm is implemented could be reused as a *software component* by a general purpose system. Second, solving problems that arise in "real world problems" usually requires a combination of methods, not only from computer algebra but also from other areas. Third, the user can use a component that runs on a high-performance computer and use the power of such a machine for a possibly application critical subtask. Since such subtasks are very often coarse grained, remote execution of them is often possible and advantageous, when it is possible to use the power of high-performance hardware on a remote execution site.

Below we will describe how we achieved such a combination of systems in the case of a high performance Gröbner bases solver which can use the power of shared-memory multiprocessors. Thus this software will perform best on special hardware and should not be ported to all platforms on which a potential user works. The encapsulation of the parallel system and access to it is done with Java using various techniques that are well supported by Java.

1.1 High Performance Gröbner Bases Computation

The computation of Gröbner bases [5] is of central importance in computer algebra. Developping algorithms and building high-performance systems for their efficient computation is a central topic of research. Our research group takes part in this efforts and has built a high-performance Gröbner solver [2]. This software that we made available via the WWW is based on the computer algebra nucleus CANNES [8]. CANNES gives a very efficient implementation of arbitrary precision integers on the basis of GNU/MP and of lists, including an automatic garbage collection. The Gröbner solver uses the parallel version PARCAN [8] of CANNES and shows good speed-ups on shared-memory multiprocessors. Even super-linear speed-ups due to strategy effects have been observed [2]. It currently runs on multi-processor SUN SPARC stations.

On the algorithmic side, improvements of the so called *Gröbner walk* technique have been developed [3, 4] and implemented. So far the most efficient of these is the *fractal walk* [1], which is the version we use in the system described in this paper.

2 Accessing the High-Performance Gröbner Bases Software via the WWW

We provide three different access methods to our high-performance symbolic software.

Methods that allow scripting access to the high-performance solvers out of a general purpose algebra system are one of them. This method of access will be described in Sec. 2.2.

A graphical user interface in the form of a Java applet will be more convenient for casual users. Using copy and paste operations data can be transferred from an application into the data input Window of the applet. The format of various algebra systems can be used for input and output to the applet. Support for other formats can be added dynamically by a user, if he provides converters to the new target system and our internally used format. The graphical user interface in this sophisticated version uses several features of JDK 1.1. We will describe it in more detail in Sec. 2.1.

To allow access to the system from older browsers we still export a simpler graphical user interface that uses JDK 1.0.2.

The graphical user interfaces can be reached from the following URL:
`http://www-sr.informatik.uni-tuebingen.de/projects/webcomponents/`
If you are interested in the software for scripting access, write an e-mail to the first author.

2.1 A Sophisticated Graphical Java Interface

For a description of the functionality of this user interface, we refer to the URL given above. From this Web page, the Java applet building the graphical user interface itself and some commenting Web pages can be reached. Below we will focus on some important points in its internal organization.

Dealing with Different Representations of Mathematical Data. Different computer algebra systems often use different syntactic representations of the mathematical objects they are dealing with. Several suggestions have been made for a common exchange format between different systems. These include the *multi-protocol* (MP) [9], OpenMath [7], or the MathBus [14]. Ideally, all systems should support one of these formats. Then the communication between different systems would be greatly facilitated. The existence of several exchange formats already indicates that no general agreement is in sight.

Our solution to this problem has been as follows. We take one of the common exchange formats for our internal representation and provide parsers and converters from and to other formats. The format we use is the one of the MathBus [14], which was especially developed as a bridge between different systems, especially computer algebra systems. Up to now we have implemented a converter from and to the computer algebra system Mathematica and the format used by the parallel Gröbner solver. Converting data form the MathBus into another format is quite easy, since the MathBus format is a quite general expression tree format. Converting data from Mathematica into the MathBus format could be done with a parser generator.[1] We used the Java Bison system [10] for this purpose. Incorporating converters for other algebra systems can be done by simply adapting the grammar of the parser for the algebra system and writing the straight-forward conversion from the MathBus format into the format of the new system. So an extension to many other formats is quite easy.

[1] The same holds for the format used by the parallel Gröbner solver.

The converter classes all inherit from a common class, and the choice of the actually used converter class is done via the dynamic class selection and load mechanism of Java. The user can choose a converter via selection from a list or by providing its name. Thus an unlimited number of new converters can be included into the graphical user interface dynamically, if the appropriate naming conventions are obeyed.

Choice of a Good Computation Strategy. The result of a Gröbner basis computation is independent of the value of several *internal parameters* but their appropriate choice usually has a dramatic influence on the computation time. Many users who need the result of a Gröbner basis computation but are not researchers in the field of Gröbner bases will find it valuable if the system provides a good heuristic choice for these internal parameters. Thus we have included the computation of such an heuristics into our applet. Having the full power of an object-oriented language such as Java available at the WWW-interface, this is a relatively easy task. However, using a simpler interface such as an HTML-form, this useful functionality would be hard to provide.

Internal Organization of the Different Computations. A schematic view of the internal organization is given in Fig. 1. The graphical user interface, the functions that select a computation strategy and the converters from the different data formats are all written in Java and will be performed on the client computer.

The following programs will be executed on our parallel server computer. The parallel Gröbner solver; a high-performance implementation of the fractal Gröbner walk; a small C program based on the SACLIB computer algebra library [6] which expands polynomials. This function serves as an improvement to the user friendliness of the interface, because the user might give polynomials in non-expanded form, but the Gröbner solver expects its input polynomials in expanded form. These programs are wrapped into Java classes. The connection between the applet residing on the client and the classes on the server side is achieved via RMI. For performance reasons the first two of these programs should always run on a high-performance computer. The last of these functions could be included into the applet. However, using an existing C-library the required functionality could be implemented with little effort in C, whereas porting the needed functionality from the SACLIB to Java would have required a much greater effort. Thus this functionality also is assigned to the server machine. Wrapping it as a Java class and making it available to the applet via RMI gave almost the same functionality as a direct Java implementation and allowed the reuse of high-quality C-code.

RMI allowed a very nice integration of this wrapped high-performance code into the Java code written for the client. Besides the interaction of the user with the GUI, the general data flow is that the input data are parsed into MathBus format according to the user specified parser. The input polynomials are normalized by the component that expands polynomials, an the result is used as input to the Gröbner solver, together with the user input of the term ordering induced by the ordering on variables specified by the user. Moreover, this function also gets the "internal parameters" as input, which were either chosen by our heuristics or specialized by the user. The return value of this function is the desired result unless a Gröbner Basis with respect to the lexicographic ordering has to be computed using the fractal Gröbner walk. In this case, the return value is the input to the Gröbner walk component, together with the parameters for start ordering and target ordering, which are known to the program.

This data flow requires several calls to the remote objects together with some minor computations in the client. The implemen-

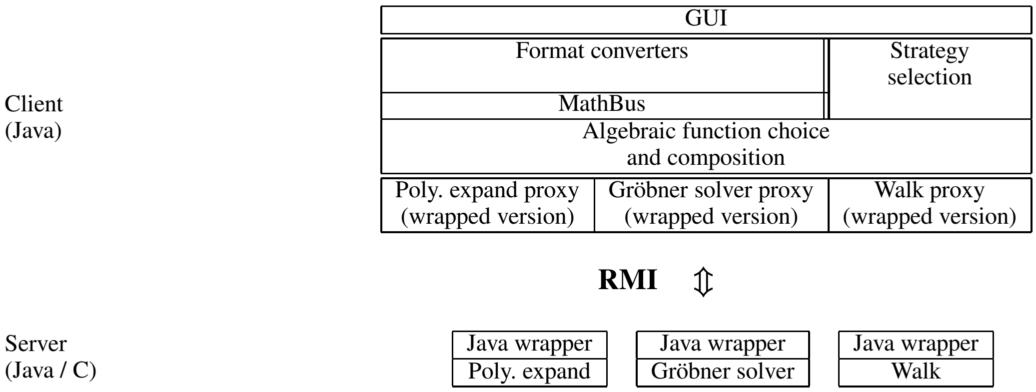

Figure 1: Schematic overview of the software components

tation of the code for these computations and the required calls to the remote objects could be written almost as if only local calls to Java objects had to be performed. Especially the treatment of exceptional states—parse errors in the converters written in Java, out of memory exceptions for the remote components or other special states such as a failure in a Gröbner basis computation—could be done uniformly by the Java exception mechanism, independent of their status as local or remote calls. For this purpose the Java wrapper for the C-programs examine and transform diagnostic output of the C-programs into Java exceptions, which then are transported transparently to the client by RMI. The output of the C-programs which constitute the result of the computation is wrapped into the return value of the Java wrapper function.

Thus this program infrastructure is well suited to be enhanced by additional software components—e. g. ones that solve a system of non-linear equations via a Gröbner basis computation. The program complexity for connecting such remote components is not higher than using local Java method calls in the implementation of a new Java method.

2.2 Scripting Access to the High-Performance Software

Although access via one of the graphical user interfaces described above is much more convenient for a casual use of the parallel software, frequent users might want to have scripting access to our parallel system out of their own algebra systems.

We thus provide Java classes that allow the various general purpose systems to encapsulate the remote access to the parallel Gröbner solver in a function of the particular system. To overcome the security restrictions on applets these classes have to be installed on the client computer. Most of the client side classes of the sophisticated graphical user interface (cf. Sec. 2.1) are reused for these purpose, and via RMI they access the identical server side software as the sophisticated graphical user interface does.

If a host algebra system allows direct access to Java, these classes can be incorporated directly. However, the current general purpose algebra systems do not have this feature. Thus the common data exchange will be over IO-streams. We have implemented a sample Java program that reads its input from standard input, writes the result to standard output, and the values of any exceptions to standard error. If the host algebra system allows socket communication, this program is easily adapted to it.

We are currently working on the implementation of functions for the computer algebra systems Mapleand Mathematica, which behave to the user in the same way as the built-in Maple resp. Mathematica functions for Gröbner basis computations do, but which will use the scripting access to our parallel Gröbner solver and the fractal Gröbner walk component to perform the computations instead of using a local sequential computation.

2.3 Use of Java Beans

Currently, we have not provided our Java components as Java Beans but as simple applets and stand alone Java programs and Java class libraries. Using Java Beans for graphical user interface had the advantage that frequent users could store their own default settings instead of using the ones provided by us. However, there are few default settings independent of the the computation of a heuristics for the choice of the internal parameters. Changes in this heuristics cannot be easily accomplished by the user by the currently available Java Beans mechanisms. Thus up to now the use of Java Beans seem to have little advantages over a simpler Java applet for our particular application.

Morevover, we feel that for frequent users the access method of choice is to use the scripting interface out of their general purpose algebra systems. Up to now the use of Java Beans gives little advantages for the scripting interface: Some of our functions need an additional parameter which could be avoided using Java Beans, but this is a minor issue in the few lines of code that has to be written once for any target algebra system.

As soon as the general purpose computer algebra systems provide direct access to Java and Java Beans, then the use of Java Beans is worth being reconsidered. Nevertheless, it takes only a relatively minor effort to transform our components to Java Beans.

Acknowledgments. We are grateful to B. Amrhein and O. Gloor for many helpful discussions about Gröbner basis computations. The software kernels that we wrapped to components and connected via Java, and made available on the WWW were to a great extend built within their research on high-performance Gröbner basis computations. This research is supported by the *Deutsche Forschungsgemeinschaft* under grant Ku 966/2-2. We are grateful to R. Zippel for many helpful discussions about the MathBus. The comments of the anonymous referees are gratefully acknowledged.

References

[1] AMRHEIN, B., AND GLOOR, O. The fractal walk. In Winkler [13].

[2] AMRHEIN, B., GLOOR, O., AND KÜCHLIN, W. A case study of multi-threaded Gröbner basis completion. In Lakshman [12].

[3] AMRHEIN, B., GLOOR, O., AND KÜCHLIN, W. Walking faster. In *Design and Implementation of Symbolic Computation Systems — International Symposium DISCO '96* (Karlsruhe, Germany, Sept. 1996), J. Calmet and C. Limongelli, Eds., vol. 1128 of *Lecture Notes in Computer Science*, Springer-Verlag.

[4] AMRHEIN, B., GLOOR, O., AND KÜCHLIN, W. On the Walk. *Theoretical Comput. Sci. 187* (1997), 179–202.

[5] BECKER, T., AND WEISPFENNING, V. *Gröbner Bases: A Computational Approach to Commutative Algebra*, vol. 141 of *Graduate Texts in Mathematics*. Springer-Verlag, 1993.

[6] BUCHBERGER, B., COLLINS, G. E., ENCARNACIÓN, M. J., HONG, H., JOHNSON, J. R., KRANDICK, W., LOOS, R., MANDACHE, A., NEUBACHER, A., AND VIELHABER, H. *SACLIB User's Guide*. Johannes Kepler Universität, 4020 Linz, Austria, Mar. 1993. Available via anonymous ftp at `melmac.risc.uni-linz.ac.at` in `pub/saclib`.

[7] DALMAS, S., GAËTANO, M., AND WATT, S. An OpenMath 1.0 implementation. In Küchlin [11], pp. 241–248.

[8] GLOOR, O., AND MÜLLER, S. PARCAN—a parallel computer algebra nucleus. In preparation, 1997.

[9] GRAY, S., KAJLER, N., AND WANG, P. MP: A protocol for efficient exchange of mathematical expressions. In *Proc. Intl. Symposium on Symbolic and Algebraic Computation (ISSAC '94)* (Oxford, July 1994), Association for Computing Machinery, pp. 330–335.

[10] HEIMBIGNER, D. *JAVA Bison—Parser and Lexer Generation for Java*. University of Colorado at Boulder, 1997. Available at `ftp://ftp.cs.colorado.edu/pub/cs/distribs/arcadia/jb.tar`.

[11] KÜCHLIN, W., Ed. *Proceedings of the 1997 International Symposium on Symbolic and Algebraic Computation (ISSAC '97)* (Aston Wailea Resort, Maui, Hawaii, 1997), The Association for Computing Machinery, ACM.

[12] LAKSHMAN, Y. N., Ed. *Proc. Intl. Symposium on Symbolic and Algebraic Computation (ISSAC '96)* (Zürich, July 1996), Association for Computing Machinery.

[13] WINKLER, F., Ed. *33 Years of Gröbner Bases* (Feb. 1998), Cambridge University Press.

[14] ZIPPEL, R. *The MathBus Term Structure*. Cornell University, Ithaca, NY 14853, U. S. A., 1997. `http://www2.cs.cornell.edu/Simlab/papers/mathbus/mathTerm.htm`.