GIANT: GRAPHICAL ALGEBRAIC NUMBER THEORY

ANEESH KARVE AND SEBASTIAN PAULI

Dedicated to Michael Pohst on his 60th Birthday

Abstract. While most algebra is done by writing text and formulas, diagrams have always been used to present structural information clearly and concisely. Text shells are the *de facto* interface for computational algebraic number theory, but they are incapable of presenting structural information graphically. We present GiANT, a newly developed graphical interface for working with number fields. GiANT offers interactive diagrams, drag-and-drop functionality, and typeset formulas.

1. Introduction

Software for number theoretical computations has evolved from stand-alone programs, to Fortran and C libraries, to shells. Shells have opened these systems to a much larger user community. Computer algebra systems are now widely used by number theorists for calculations and experimentation.

At the same time, the tasks that can be solved in computational number theory have become more complex. The focus has changed from the computation of invariants such as integral basis, unit groups, class groups, and Galois groups, to computations in areas such as class field theory, where more than one number field is considered at a time. Diagrams offer a useful overview of the relations between fields, but these are typically done on paper since most systems specialized for number theory cannot visually display such relations. A graphical interface to display and manipulate diagrams, drag and drop elements, and perform common operations could be of great benefit to the user. In addition, mathematical typesetting could improve the representation of formulas.

Several general computer algebra systems, such as Maple [Ma05] and Mathematica [Wo05], provide graphical interfaces. They offer typesetting and plotting, but they do not allow graphical manipulation of algebraic objects. For group theory the XGAP [CN04] package for GAP [GAP] offers a tool for viewing and manipulating subgroup lattices and other structural information on UNIX systems running X Windows. It allows bidirectional communication between the interface and GAP, though this is not automatic. Working with elements of groups is not supported.

We present GiANT, a platform independent graphical interface to the KANT shell KASH [DF+97], a computer algebra system for number theory. Number fields are displayed and related to each other on the GiANT desktop. GiANT allows the user to work with the invariants of the number fields and with polynomials, ideals, and elements. Many operations between these can be conducted in GiANT, and mathematical typesetting is used for clarity of representation.

2. GiANT

In the following sections we highlight key features of GiANT.

Aneesh Karve was generously supported by the German Academic Exchange Service (http://www.daad.org) during the creation of GiANT..

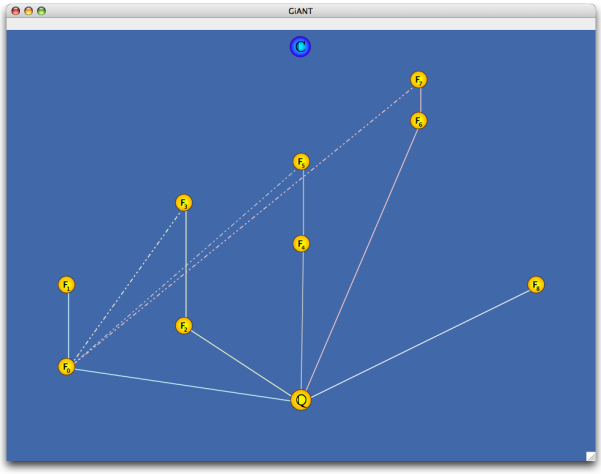

FIGURE 1. GiANT's interactive desktop displays towers of number fields.

The Tower of Fields. GiANT opens with a blank desktop. Above the desktop is a menu system to create and manipulate number fields. As the user creates number fields, they appear on the desktop as icons, and are arranged into towers (FIGURE 1). Hovering over a field icon with the mouse displays the field's generating polynomial. Subfield relationships within a tower are indicated with solid lines, and in neighboring towers with dotted lines. For visual clarity a different line color is used for each tower.

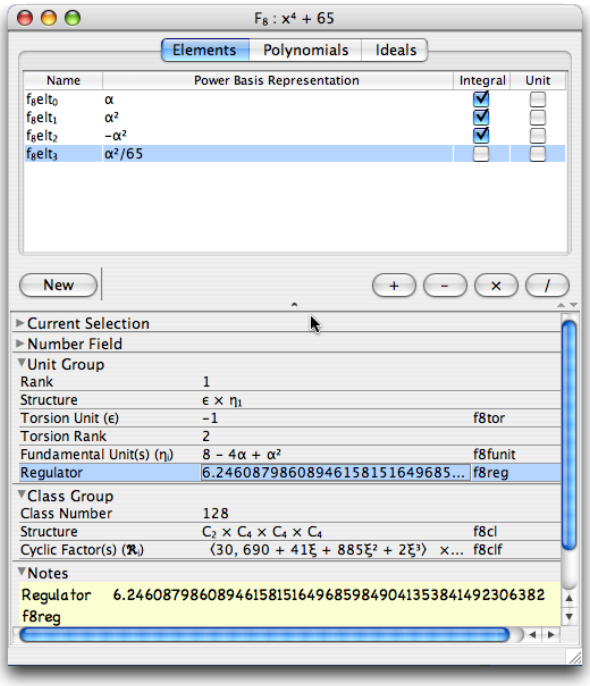

FIGURE 2. The Inspector Window displays information for a particular number field.

Working with Fields: The Inspector Window. Clicking on a field icon opens its Inspector Window (FIGURE 2). The window title bar displays the name and generating polynomial of the field. The upper half of the window consists of three tabs: *Elements*, *Polynomials*, and *Ideals*. Under each tab is a table containing variables of the same type. Columns in the table include the variable name, its power basis representation, and other key properties.

Below each table are the *New* button, which is used to create new variables by entering expressions, and four buttons for arithmetic operations. The operations can be applied by selecting operands from the table. The result appears automatically as a new variable in the table.

The lower half of the Inspector Window contains four sections: *Current Selection, Number Field, Unit Group, Class Group,* and *Notes*. Each section can be collapsed or expanded as needed.

Current Selection displays invariants for one or more variables. Taking field elements as an example, *Current Selection* displays the norm, trace, minimal polynomial, and integral basis representation for any elements currently selected in the *Elements* table. *Number Field* displays invariants for the entire field. *Unit Group* displays the rank, structure, torsion unit, fundamental units, and regulator. *Class Group* displays the class number, the structure of the class group, and the generators of its cyclic factors. *Notes* is a freely editable text box.

Computations which may take more than a few seconds to complete, such as determining the class group, execute only when the user requests them. This avoids unnecessary latency.

Automatic Variable Naming and Abbreviations. For elements, polynomials, and ideals, GiANT generates automatic names by combining the field name, variable type, and variable index into a single string. For example, the first element the user creates in the field named `f0` is automatically named `f0elt0`. This makes it easy to identify variables generated by GiANT, and to use them in the corresponding KASH session. Though GiANT always offers automatic variable names, the user can provide custom names. In addition to user-created variables, most auto-computed invariants are also named by GiANT.

GiANT allows the use of single-letter abbreviations for commonly used values, such as primitive elements, to facilitate the input of expressions. For simplicity, abbreviations are consistent across number fields. GiANT translates abbreviations into globally bound variable names which the underlying shell understands. Such abbreviations are not possible in a shell-only environment, since a shell has no notion of which number field the user is working with.

Drag-And-Drop Convenience. GiANT employs drag-and-drop for convenient moving of elements between algebraic structures and for the execution of other operations. When working with an element in a number field, for example, we may wish to study its minimal polynomial. The GiANT user simply drags the element from its table and drops it onto the *Polynomials* tab. The *Polynomials* tab then opens, adds the minimal polynomial to its table and highlights it. Similarly, the user can drop one or two elements onto the *Ideals* tab to generate a new ideal. It is even possible to create relative field extensions by dropping irreducible polynomials from a ground field onto the desktop. The user is then free to move elements between the ground field and the extension by dragging and dropping.

The Global Inspector. The Inspector Window discussed above organizes all of the variables in a single number field. Nevertheless, users may wish to simultaneously view *all* variables they have created, irrespective of type or parent field, in a single window.

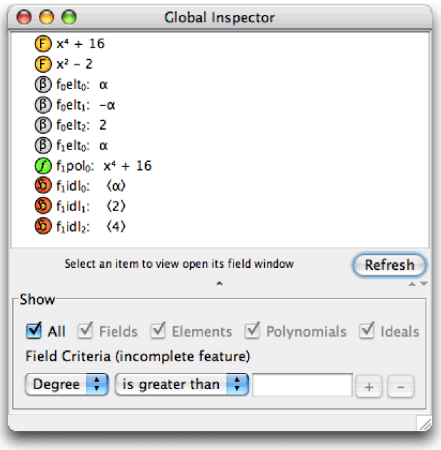

FIGURE 3. The global inspector displays all variables the user has created.

The Global Inspector (FIGURE 3) makes this possible. Number fields, elements, polynomials, and ideals — or any subset of these four variable types — are displayed in a single list. Selecting a variable from the list brings forward the Inspector Window where this variable resides; the user can then manipulate it. In addition the Global Inspector allows the user to specify filtering criteria which variables must meet in order to be displayed in the list.

The Shell Behind GiANT. GiANT uses the KANT shell KASH to perform computations. The user can work directly with the shell by selecting *Show KASH* from the *View* menu. This displays a standard KASH console. Any variables created with the graphical interface are also available in the console. The user may use the console to access features of KASH which are not graphically available. Alternatively, GiANT can be used as a scripting tool for KASH since all graphically-driven activities generate KASH code.

3. CONCLUSION

The motivation behind GiANT was to provide intuitive access to the functionality of KASH. To be sure, we have not completely succeeded in this regard, but nevertheless hope to have offered some ideas in the right direction. In its present state of development GiANT can be used for teaching and presentations. To use GiANT for research in algebraic number theory the following issues should be addressed. A bidirectional integration of graphical manipulation and a classic text-based shell should be achieved such that objects, and structural information about them, are displayed graphically as these objects are created in the shell. In GiANT the objects that are generated in the graphical user interface are available in the shell, but objects generated in the shell do not appear in the graphical interface.

A more open software architecture would make it possible to use GiANT for other computer algebra systems and for algebraic structures other than number fields. Said architecture should be designed in such a way that the same methods that are used to display lattices of subfields are used, for example, to display lattices of subgroups.

Because algebra is mostly represented in formulas, a system that graphically displays structural information and allows graphical as well as command line driven manipulation of structures and elements would be the ideal solution. This would take the best from both worlds. On the one hand complex tasks are more easily

done on the command line. On the other hand routine tasks, structural information, and an overview of defined objects are best done graphically.

GiANT is written in Java 1.4. For further research and development the GiANT source code is made available under the GNU General Public License (GPL). GiANT can be found on the worldwide web at

http://giantsystem.sourceforge.net.

4. Acknowledgements

The authors would like to wish Michael Pohst a happy 60$^{\text{th}}$ birthday. We have had the pleasure of working with him at Technische Universität Berlin. Herr Pohst is not only an extraordinary mathematician, but a kind human being as well.

Special thanks to Claus Fieker, Sebastian Freundt, Jürgen Klüners, Mike Lache, and Osmanbey Uzunkol for valuable feedback on the development and use of GiANT. Mike also critiqued an early draft of this manuscript; Frances Clerk helped bring it into its final form.

References

[DF$^+$97] M. Daberkow, C. Fieker, J. Klüners, M. Pohst, K. Roegner, M. Schörnig, and K. Wildanger, KANT V4, J. Symb. Comp. **11** (1997), 267–283, http://www.math.tu-berlin.de/~kant/.

[CN04] F. Celler, M. Neunhöffer, *GAP package XGAP – a graphical user interface for GAP*, 2004, http://www-gap.mcs.st-and.ac.uk/Packages/xgap.html

[GAP] *GAP - Groups, Algorithms, Programming - a System for Computational Discrete Algebra*, 2005, http://www.gap-system.org

[Ma05] Maplesoft, *Maple*, 2005, http://www.maplesoft.com

[Wo05] Wolfram Research, *Mathematica*, 2005, http://www.wolfram.com

Aneesh Karve, Department of Computer Sciences, University of Wisconsin-Madison, 1210 West Dayton Street, Madison, WI 53706-1685, USA

E-mail address: karve@cs.wisc.edu

Sebastian Pauli, Institut für Mathematik, MA 8–1, Technische Universität Berlin, Strasse des 17. Juni 136, 10623 Berlin, Germany

E-mail address: pauli@math.tu-berlin.de