

Automatically Exploring the Domain of Residue Classes

Extended Abstract

Andreas Meier and Volker Sorge

Fachbereich Informatik

Universität des Saarlandes

D-66041 Saarbrücken, Germany

{ameier|sorge}@ags.uni-sb.de

<http://www.ags.uni-sb.de/~{ameier|sorge}>

1 Introduction

We describe a module for exploring simple algebraic properties of operations on residue class sets over the integers. The framework is implemented within the Ω MEGA theorem proving environment [1]. It employs computations of the computer algebra system GAP [3] to classify a given residue class set together with one or two operations in terms of its algebraic structure. During this classification process proof obligations for proving or refuting single properties are generated. These proof obligations are passed to Ω MEGA's multi-strategy proof planner MULTI [5] that constructs a proof with the help of GAP and MAPLE [7].

Since the presented exploration module has originated from work done in the context of tutor systems, the motivation is not to obtain new results in finite algebra. It shall rather enable a user to learn fundamental algebraic notions by fiddling about with arbitrary residue class sets and combinations of operations. Moreover the module enables the automatic exploration of large testbeds.

In our context a residue class set over the integers is either the set of all congruence classes modulo an integer n , i.e., \mathbb{Z}_n , or an arbitrary subset of \mathbb{Z}_n . Some examples are \mathbb{Z}_3 , \mathbb{Z}_5 , $\mathbb{Z}_3 \setminus \{\bar{1}_3\}$, $\mathbb{Z}_5 \setminus \{\bar{0}_5\}$, $\{\bar{1}_6, \bar{3}_6, \bar{5}_6\}, \dots$ where $\bar{1}_3$ denotes the congruence class 1 modulo 3. An operation on a residue class set is an arbitrary combination of the constant operations, addition, multiplication, and subtraction on congruence classes.

2 The Exploration Module

The idea of the exploration module is to classify a residue class set with one operation in terms of the algebraic structures magma, semi-group, quasi-group, monoid, loop, or group. A set with two given operations is categorized in terms of ring, ring-with-one, division ring, or field. The module checks properties of a

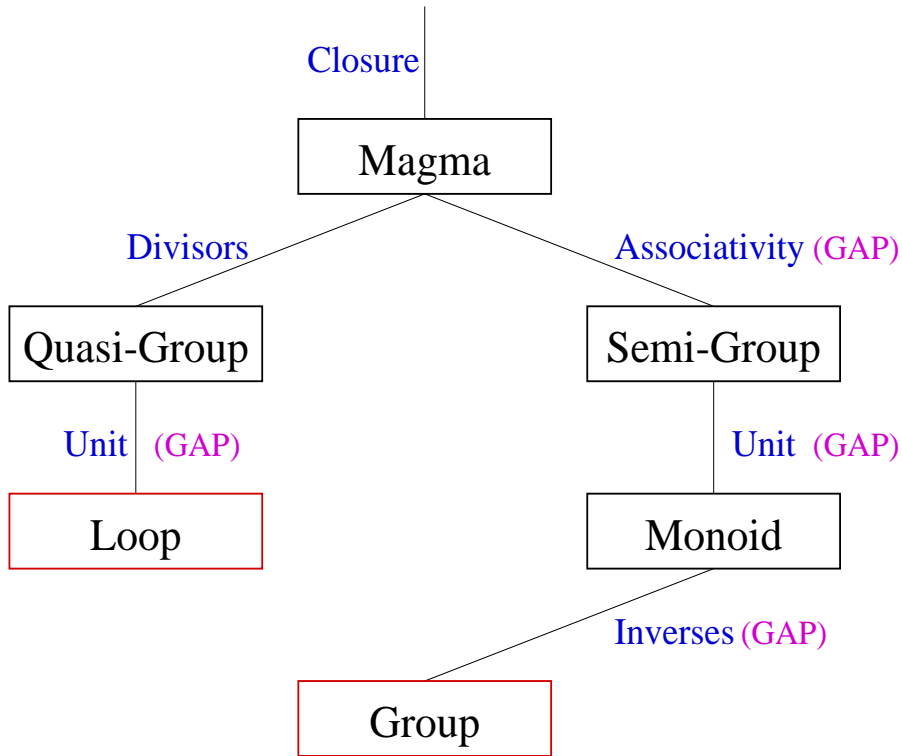


Figure 1: The basic exploration cycle.

given set automatically in order to eventually classify it as one of these algebraic structures.

The input to the exploration module is a set and either one or two corresponding operations. The basic exploration cycle is depicted in Fig. 1. In detail it proceeds as follows: The exploration module checks first whether the given structure forms at least a magma. If not it stops. Otherwise it checks associativity to see if we have a semi-group. If this holds, the module checks if there exists a unit element, i.e., whether we have a monoid. Finally, if inverses exist we have a group. In case the exploration module cannot establish associativity it immediately proceeds to check the existence of divisors for each two elements of the set which corresponds to the axiom of quasi-groups. Should we have a quasi-group, we check for the existence of a unit element, to test whether we have a loop. The exploration will then stop in any case since we already know that the given algebra does not form a group (since it is not associative). After the given set is classified as described, the module always checks whether the given operation is commutative.

The exploration module is implemented within the Ω MEGA environment and incorporates computations of the computer algebra system GAP. Its main routines first constructs a multiplication table with respect to the given set and operation. The table is employed to check the closure property and to

check the existence of divisors, i.e., the axiom for quasi-groups. Furthermore, if the computed multiplication table is closed under the respective operation it is used to construct the appropriate magma in GAP. GAP can then be employed to query for associativity and commutativity as well as to compute the unit element and inverses for single elements.

All query functions of the exploration module return useful results in both the positive and the negative case: That is, for instance, if GAP can compute a unit element for a given magma this element is returned. In case GAP fails to find a unit element the multiplication table is used to determine a set of elements that suffice to refute the existence of a unit element for the given magma. A special case is the failure of the query for associativity, since then we try to use MAPLE to compute a particular solution for the associativity equation. If such a non-general solution exists it is exploited to determine a triple of elements for which associativity does not hold.

When checking each property the exploration module generates a theorem according to the obtained result. For instance, if the module establishes that associativity holds for the given set, the respective theorem is created. In case of failure the negation of the theorem is produced. The generated theorems are passed to Ω MEGA's multi-strategy proof planner and proved as briefly sketched in the next section.

In case the exploration module is called with two distinct operations, we first execute the above classification process for the set together with the first operation. Should the structure form an abelian group the exploration process commences for the set reduced by the unit element of the first operation with respect to the second operation. If this structure is at least a magma we try to establish the distributivity between the two operations. This is done again using GAP. If successful we can classify the set together with the two operations, depending on the classification of the second structure: we have a ring, if it is a magma or semi-group, a ring-with-one if it is a monoid, a division ring if it is a group, and a field if it is additionally abelian.

3 Generating Proofs

Proof obligations generated by the exploration module are passed to Ω MEGA's multi-strategy proof planner MULTI. MULTI constructs a proof with the help of the two computer algebra systems MAPLE and GAP. The proofs are relatively straightforward and do not pose a very hard problem for the planner. Therefore we can be relatively sure to successfully generate the necessary proofs.

Proof planning [2] considers mathematical theorems as planning problems where an *initial partial plan* is composed of the proof *assumptions* and the theorem as *open goal*. A proof is then constructed with the help of abstract planning steps, called *methods*, that are essentially partial specifications of tactics known from tactical theorem proving. In the Ω MEGA system the traditional proof planning approach is enriched by incorporating mathematical knowledge into the planning process (see [6] for details). That is, methods can encode general proving steps as well as knowledge particular to a mathematical domain.

Moreover, *control rules* specify how to traverse the search space by preferring, rejecting, or enforcing the application of methods in certain domains or proof situations. The multi-strategy proof planner MULTI allows also for the specification of different planning strategies to control the overall planning behavior. In the domain of residue classes we employ three strategies to prove occurring theorems which we will briefly sketch (for a more detailed account see [4]).

The motivation for the first strategy is to implement a rather naïve approach of proving a property of a residue class set. It proceeds by rewriting statements on residue classes into corresponding statements on integers, especially by transforming the residue class set into a set of corresponding integers. It then exhaustively checks all possible combinations of these integers with respect to the property that is to be proved or refuted. The strategy proceeds in two different ways, depending on whether (1) a universal or (2) an existential property has to be proved. In case (1) a split over all the elements in the set involved is performed and the property is proved for every single element separately. In case (2) the single elements of the set involved are examined until one is found for which the property in question holds. To prune the search space the strategy employs a control rule that exploits the same functionality as used in the exploration module. This enables to compute directly the witness terms for existentially quantified variables with the help of GAP.

The aim of the second strategy is to use as much as possible equational reasoning to prove properties of residue classes. Similarly to the first strategy it converts statements on residue classes into corresponding statements on integers. But instead of checking the validity of the statements for all possible cases, it tries to solve occurring equations in a general way. The strategy uses a method that can justify an equational goal using MAPLE and thereby might instantiate meta-variables¹. In detail, it works as follows: if an open goal is an equation MAPLE's function `solve` is applied to check whether the equality actually holds. If the equation contains meta-variables these are considered as the variables the equation is to be solved for and they are supplied to `solve` as additional arguments. In case the equation involves modulo functions with the same factor on both sides MAPLE's function `msolve` is used, instead.

Contrary to the first two strategies whose idea is to reduce expressions on residue classes to expressions on numbers, the last strategy tries to tackle new problems by applying already known theorems. Theorems are stored in Ω MEGA's theory database and, in order to keep the number of possible theorems small, only those in the theory of residue classes are eligible for application.

From the three strategies only the very first can be applied to all given problems whereas the other two might fail, for instance in the case when MAPLE does not return a useful result (e.g., a term involving a rational number) or the necessary theorems to solve a problem are not given in the database. On the other hand, the latter two strategies are relatively efficient, since they are independent of the size of the residue class set involved, whereas the exhaustive case analysis can become quite lengthy for large sets. Thus, MULTI always

¹Meta-variables are dummy variables for witness terms introduced for existentially quantified variables. They have to be instantiated during the proof planning process.

tries to employ the strategies for equational reasoning and application of known theorems first. Only when these fail MULTI applies the exhaustive case analysis.

4 Extensions and Further Work

We are currently extending the exploration module to be able to automatically detect isomorphic structures within a given collection of residue class sets. In case we can establish that two distinct structures form a group we can use GAP's functionality to compute an isomorphism if one exists. The proof is then again constructed by MULTI. However, in case the structures in question do not form a group we can so far only test isomorphism by trying to prove or refute the property directly with the proof planner which is not always successful.

Future work will include the extension of the exploration module to deal with more general algebraic sets than residue classes. Since one necessary prerequisite is that the sets involved are finite and operations are computable, we are considering to introduce functionality for dealing with sets of permutations and matrices over finite fields.

References

- [1] The Ω MEGA Group Ω Mega: Towards a Mathematical Assistant. In W. McCune, editor, *Proceedings of CADE-14*, volume 1249 of *LNAI*, pages 252–255, Townsville, Australia, 1997. Springer Verlag, Berlin, Germany.
- [2] A. Bundy. The Use of Explicit Plans to Guide Inductive Proofs. In E. Lusk and R. Overbeek, editors, *Proceedings of CADE-9*, volume 310 of *LNCS*, Argonne, IL, USA, 1988. Springer Verlag, Berlin, Germany.
- [3] The GAP Group, Aachen, St Andrews. *GAP - Groups, Algorithms, and Programming, Ver. 4*, 1998. <http://www-gap.dcs.st-and.ac.uk/~gap>.
- [4] A. Meier and V. Sorge. Exploring properties of residue classes. In M. Kerber and M. Kohlhase, editors, *Proceedings of Calculemus-2000*, St. Andrews, 6–7 August 2000. AK Peters, New York, NY, USA. forthcoming.
- [5] E. Melis and A. Meier. Proof planning with multiple strategies. In *Proc. of the First International Conference on Computational Logic*, London, United Kingdom, 2000. Springer Verlag, Berlin, Germany.
- [6] E. Melis and J. Siekmann. Knowledge-based proof planning. *Artificial Intelligence*, 1999.
- [7] Darren Redfern. *The Maple Handbook: Maple V Release 5*. Springer Verlag, Berlin, Germany, 1999.