Automated Parameterisation of Finite Algebras

Simon Colton[1] and Volker Sorge[2]

[1] Department of Computing
Imperial College, London, UK
sgc@doc.ic.ac.uk
[2] School of Computer Science
University of Birmingham, UK
V.Sorge@cs.bham.ac.uk

Abstract. In previous work we have designed an automatic bootstrapping algorithm to classify finite algebraic structures by generating properties that uniquely describe and discriminate different equivalence classes. One of the drawbacks of the approach was that during the classification a large number different discriminating properties were generated. This made it particularly difficult to compare classifying properties for different sizes of algebraic structures. To minimise the overall number of properties needed we have now experimented with parameterising structures by counting the number of elements with particular properties. Isomorphism classes are then discriminated by the different number of elements with the same property. With this new approach we can now classify large numbers of algebraic structures using only a small number of properties. We were able to construct and prove parameterisations for algebraic structures like loops of size 6 and groups of size 8. However, the approach is currently limited as translating counting arguments into pure first order or propositional logic, often makes for prohibitively long problem formulations.

1 Introduction

In previous work, we have automated the construction of fully verified qualitative classification theorems for finite algebraic structures of a given size, satisfying a given axiom set. For instance, given the axioms of group theory and specification of size 6, our system would: invent the concepts of Abelian and non-Abelian groups; prove that these concepts classify groups of size 6 into isomorphism classes; and prove that no other isomorphism classes exist. We use a bootstrapping technique which involves the combination of many automated reasoning systems, including theorem provers, model generators, SAT solvers, machine learning and computer algebra systems. Without going into detail (see [3] for this), a major part of the bootstrapping algorithm is to take two non-equivalent examples of the algebra and find an invariant property that one has and the other doesn't, which proves that they cannot be equivalent. When such a property is found, a branching point on a classification tree is produced. The leaves of the final tree represent concepts which define equivalence classes. Using

this approach, we have been able to construct the first qualitative classification theorems for loops and quasigroups up to both isomorphism [3] and isotopism [5], e.g., for loops of size 6, which have 109 isomorphism classes and 22 isotopism classes.[1]

While this approach was successful, there are three main drawbacks to continuing in this way, namely (i) the number of equivalence classes (both isomorphism and isotopism) explodes as the size of the algebra increases (e.g., there are more than 100 million loops of size 8 up to isomorphism), (ii) the reasoning tools we use have difficulty with certain required theorems about algebras of size 8 or more, and (iii) the classification trees produced for larger sizes bear little resemblance to those for smaller sizes for a given axiomatisation. With respect to this final point, we have considered data-mining the trees to find concepts which appear in multiple classification trees. However, it seems more sensible to first experiment with more proactive approaches to generating classification results, by initially imposing a tree structure which uses common concepts, before filling in the rest of the tree.

Drawing on existing mathematical results, we note that there are 14 groups of size 8 or smaller up to isomorphism. Moreover, they are usually classified either in terms of a parameterisation consisting of a family that they belong to and their size, e.g., the cyclic group of order 5 (C_5), the dihedral group of order 8 (D_4), etc., or in terms of a cross product of such parameterised groups, e.g., $C_4 \times C_4$. While we have performed some experiments with cross products, we concentrate here on the automatic generation of parameterisations of finite algebraic structures. Our first approach has been to look at parameterisations of algebraic structures in terms of a list of set sizes, where each set contains elements of the structures with particular properties. For instance, groups up to size 6 can be classified up to isomorphism by using a parameterisation in terms of two coefficients: the number of elements and the number of self-inverse elements (x s.t. $x = x^{-1}$).

This choice is motivated by the desire to use counting, as this is an important tool in producing classification results, yet has been missing from our previous approaches. Moreover, we have chosen to focus our first experiments on concepts which count the number of elements of a particular type because there is a standard – if cumbersome – way of formalising such set-size results in first order logic, which enables us to get proofs of our results from automated theorem provers. In sections §2 and §3, we describe how we generated and then proved parameterisations of loops, groups and quasigroups. In §5, we use the results from these experiments to suggest improvements to our bootstrapping approach.

[1] A quasigroup is generally a non-associative structure with a Latin square property. Loops have in addition a unit element. Their quantitative classifications, i.e. the number of different equivalence classes, have been known for some time [6].

2 Generating Classifying Parameterisations

Suppose we start with a set of algebraic structures $A = \{A_1, \ldots, A_n\}$ and a list of *element-type concepts* $C = \{c_1, \ldots, c_k\}$. An element-type concept is a boolean test on an element in an algebraic structure, for instance whether the element is idempotent ($x * x = x$). We then define the profile of a given $a \in A$ with respect to C as: $P(a) = \langle |\{x \in a : c_1(x)\}|, \ldots, |\{x \in a : c_k(x)\}| \rangle$. We further say that C represents an *element-type parameterisation* of A if no pair of algebraic structures in A have the same profile. If A contains representatives of each isomorphism class up to a certain size n for a specific algebraic structure, then the parameterisation can be used to classify that structure up to size n, and this classification can be proved (see next section).

We constructed such classifying parameterisations for loops up to size 5, groups up to size 8 and quasigroups up to size 4 as follows (for clarity, we will use the groups up to size 8 as an illustrative example). We started with a set of groups, A, with each member being a representative of a different isomorphism class, and all the isomorphism classes covered. We used A in the background knowledge for the HR automated theory formation system. Details of how HR works can be found in [2]. For our purposes here, HR is a concept generator, i.e., given some background concepts such as the multiplication operator in groups, HR will invent concepts such as commutativity, etc. In particular, HR is able to generate hundreds of element-type concepts.

We ran HR for 1000 theory formation steps. From the resulting theory, we extracted the set, C, of element-type concepts and we used these to automatically construct a parameterisation P as follows: The first concept in the parameterisation list is chosen as the overall size of the algebraic structure, largely for reasons of comprehensibility. We then check the parameterisation against A, and remove from A any structures a for which the profile of a is different from all the others. We then iteratively add to P the concept $c \in C$ which differentiates the largest number of pairs of structures from A. Note that we say c differentiates a_1 and a_2 iff $|\{x \in a_1 : c(x)\}| \neq |\{x \in a_2 : c(x)\}|$. Each time a new concept is added, P is checked against A, and – as before – any structure which has a unique profile is removed. This iteration continues until either A is empty (in which case a full parameterisation has been constructed), or there are no concepts left to try. In the output, each structure a is presented with only the concepts needed to distinguish it from the others. The conjunction of this set of concepts is a classifying concept for the isomorphism class represented by a.

We ran the same experiment for loops up to size 5. For quasigroups up to size 4, we increased the number of theory formation steps to 10000, and for loops up to size 6, we increased it to 40000. The results are presented in table 1.

We see that the method was able to produce full parameterisations in a reasonable time (on a 2.1GHz machine) for the groups, quasigroups and loops to size 5 datasets. However, it only achieved a partial classification of 86 of the 120 loop classes up to size 6. Note the the *Element-types* column above describes the total number of element-types produced by HR, while the *Classifiers* columns describes the number of these which were used in the parameterisation. The

Domain	Max size	Classes	Achieved	Steps	Element-types	Classifiers	Time (s)
Groups	8	14	14	1000	32	3	28
Loops	5	11	11	1000	32	3	10
Loops	6	120	86	40000	736	11	2903
Quasigroups	4	42	42	10000	523	5	215

Table 1. Details of parameterisations achieved in loop, group and quasigroup theory

group theory parameterisation was particularly simple, in terms of counting 3 elements types, namely (i) elements themselves (ii) self-inverse elements and (iii) elements which appear on the diagonal of the multiplication table. The sizes 1 to 5 loop theory parameterisation also required counting only 3 element types: (a) elements themselves (b) elements on the diagonal of the multiplication table and (c) elements, x such that $\exists\, y\ (y * x = id \wedge y * y = x)$. We think it is an achievement to be able to classify all 42 quasigroups up to size 4 by counting only 5 element types, and to classify 86 of the 120 loops up to size 6 by counting 11 element types.

3 Proving Isomorphism Classes

As we saw above, for each algebraic structure satisfying a set of axioms $\mathcal{P}$, HR produces a conjunction of concepts which can be used to classify the structure. Each concept expresses a boolean test of the structure, namely whether the number of elements with a particular logic property is a particular number. For example, HR returns the cyclic group G of order 4 as the multiplication table given below, together with the axioms of group theory and the single property $2 : x^{-1} = x$, i.e., that there are exactly two self-inverse elements in G.

To prove that the conjunctions of set sizes represent classifying concepts, we reuse some of the technology implemented for the original bootstrapping algorithm as presented in [3] and its adaptations to work with satisfiability modulo theories solvers detailed in [4].

G	a	b	c	d
a	a	b	c	d
b	b	a	d	c
c	c	d	b	a
d	d	c	a	b

In a first step, we translate the set-size properties into full first order logic by expressing the counting argument in a formal way. For instance, in our example we define a property P as

$$\exists x, y_{\blacksquare} x \neq y \wedge x^{-1} = x \wedge y^{-1} = y \wedge (\forall z_{\blacksquare} z^{-1} = z \rightarrow (z = x \vee z = y)).$$

We then need to prove two types of problems: (1) proving that the given conjunction of set-size properties is an invariant under isomorphism for a particular type of algebraic structure, regardless of the size of the structures, and

therefore serves as a discriminant, and (2) that the discriminant uniquely defines an isomorphism class for algebraic structures of a given size.

Problems of type (1) are easy to formalise as $\forall A_1, A_2 . \mathcal{P}(A_1) \wedge \mathcal{P}(A_2) \wedge P(A_1) \wedge \neg P(A_2) \rightarrow A_1 \ncong A_2$. They can be expressed in first order logic by considering the sets A_1 and A_2 as arbitrary but different constants and formulating their axiomatisations with disparate operations. Proving these theorems is relatively easy and we used the first order prover Spass [7]. Problems of type (2) are less trivial since they are essentially second order problems: we have to show that all algebraic structures that have property P are also isomorphic to the representant, in our case G, or more formally:

$$\forall A . [\mathcal{P}(A) \wedge P(A)] \rightarrow [\exists \phi . \text{bijective}(\phi) \wedge \text{homomorphic}(\phi) \wedge \phi(A) = G].$$

However, since we are in a finite domain, we can explicitly formulate the problem in propositional logic: We give G in terms of its elements and multiplication table and then formulate all possible bijective mappings from an arbitrary structure A onto the elements of G. However, since the number of mappings to consider is $n!$, where n is the size of the structures A and G, the technique quickly becomes infeasible, even for small n. We therefore use a computer algebra device by restricting the mappings to consider a generating system of G, i.e., a set of elements that can generate all other elements of the structure together with all generating equations. In our example, G is the cyclic group of size 4, and thus the generating system is of the form: $\langle \{c\}, \{a = ((c * c) * c), b = c * c, c = c, d = (c * c) * c\} \rangle$. The number of bijective mappings to consider is then 4 instead of $4! = 24$. (For a detailed discussion of these techniques see [4].) While the problem formulation can still be relatively lengthy, we found that we could solve problems up to size 8 using CVC-3 [1].

4 Results and Limitations

In our experiments, we were successful in fully automatically generating and proving the necessary theorems for quasigroups of up to size 4, loops up to size 5 (as presented in §2), and groups up to size 8. We could also show the 86 loops 6 problems for which HR could return parameterisations. The most challenging bit when solving these problems was to produce the problem formulations due to their size. However, once formulations could be produced they were shown to be valid by CVC-3 in less than 1 minute.

Despite several optimisations to our routines to more efficiently produce larger problem formalisations, it is currently infeasible to even generate problems beyond those for size 8 groups and size 6 loops. This has essentially two reasons:

Firstly, we use a naive formalisation for counting arguments in first order logic. For example, one of the properties that uniquely determines one of the loops 6 isomorphism classes is that there exists exactly three elements b, such that

$$\exists\, c, d, e\ (d * d = c \wedge \neg((\exists\, f\ (f * f = d))) \wedge b * b = e \wedge e * e = c) \qquad (1)$$

To express this argument in standard first order logic we have to explicitly state that (a) there exist three elements that have property (1), (b) these three elements are all different, and (c) there do not exist any other elements with property (1). The latter is formulated in first order logic by stating that all other elements that have property (1) have to be equal to one of the three original ones. Thus (1) becomes the following larger formula:

$$\begin{aligned}\exists\, x_0, x_1, x_2 \quad & (x_0 \neq x_1) \wedge (x_0 \neq x_2) \wedge (x_1 \neq x_2) \\ & \wedge \exists\, c, d, e\ (d * d = c \wedge \neg((\exists\, f\ (f * f = d))) \wedge x_0 * x_0 = e \wedge e * e = c) \\ & \wedge \exists\, c, d, e\ (d * d = c \wedge \neg((\exists\, f\ (f * f = d))) \wedge x_1 * x_1 = e \wedge e * e = c) \\ & \wedge \exists\, c, d, e\ (d * d = c \wedge \neg((\exists\, f\ (f * f = d))) \wedge x_2 * x_2 = e \wedge e * e = c) \\ & \wedge \forall\, y\, (\exists\, c, d, e\, (d * d = c \wedge \neg((\exists\, f\, (f * f = d))) \wedge y * y = e \wedge e * e = c) \\ & \qquad \rightarrow (y = x_0) \wedge (y = x_1) \wedge (y = x_2))\end{aligned} \tag{2}$$

The resulting problems are beyond the power of current first order systems, even for algebraic structures of small size (cf. [4]). As alternative we therefore employ SMT solvers, that can deal much more effectively with large problems in finite domains. However, they generally require ground formulas as input. While they can be simply generated by eliminating the quantifiers over the finite domain, formulas and problems become very quickly unwieldily large.

This problem can be partially overcome as CVC-3 allows for problem formulations involving quantifications, where the quantifiers are restricted over a type that represents the finite set of our algebraic structure. Unfortunately, CVC-3's calculus is incomplete for quantified input formulas, which often leads to a non-conclusive result when given problem formulations containing quantifiers. Thus the only alternative is again to supply the quantifier free problem. But, for example, eliminating the quantifiers of formula (2) over a domain of six elements leads to a formula that is of size 61 MB. And while CVC-3 can deal with problems of several hundred MB in size, generating the actual input becomes quickly infeasible. We therefore currently adopt the strategy to only generate fully grounded problems if CVC-3 can not find a proof given the quantified problem. Unfortunately our experience so far shows that for larger problems CVC-3 only rarely succeeds on the quantified problems.

5 Conclusions and Future Work

We have shown that it is possible to both derive and prove novel element-type classifying parameterisations of algebraic domains. These classifications have an advantage of simplicity and homogeneity over our previously constructed classifications, because they require understanding of only a handful of element-type concepts. Also, the classification of smaller algebraic structures share concepts with those of larger structures: another improvement on our previous results.

This work suggest the following improvement to our existing bootstrapping method [3]. Given an axiomatisation of an algebraic domain, A, in one session,

we will attempt to build a classification tree for size 1 algebras, then size 2, and so on, until computational limits are reached. The method should proceed as usual, but only invent invariants which count the size of sets of elements. Furthermore, suppose that every time the tree for algebras of size n is produced, the system adds to a global list, I, all the new invariant properties it has needed to invent to achieve the classification. Then, before attempting to produce the classification tree for size $n+1$, the system should pro-actively check to see whether any p and k such that $p \in I$ and $k \in \{0, \ldots, n+1\}$ is a classifying concept, i.e., the concept of an algebraic structure, S, being such that $|\{x \in S : p(x)\}| = k$ uniquely classifies structures of type A and size $n+1$. The pairs (p, k) for which this is true will form the basis of the classification tree for size $n+1$, while for every pair (p, k) for which this is not true, the system could see whether a conjunction of such pairs provides a classifying concept, then triples, and so on.

So far we have used the bootstrapping algorithm only to show each of our parameterisations forms describes indeed an isomorphism class for the algebraic structures of that particular size. That is, we start the algorithm given the parameterisation, and it terminates returning a single equivalence class. However, we have not yet integrated the use of parameterisation into the general bootstrapping process, which will be our next step. If we can also prove results about cross products of algebras, we hope to produce trees which not only perform the classification, but which are simpler and more in line with those produced by mathematicians. In particular, we hope to discover *families* of loops and quasigroups, where a family is essentially a parameterisation such as those described above and a way of choosing parameters which guarantees that the resulting concept will describe an isomorphism class for all sizes. In this fashion, it is not impossible that automated classification tools could begin to have an impact on research mathematics.

References

1. C Barrett and S Berezin. CVC Lite: A new implementation of the cooperating validity checker. In *Computer Aided Verification, 16^{th} Int. Conf., 2004*.
2. S Colton. *Automated Theory Formation in Pure Mathematics*. Springer, 2002.
3. S Colton, A Meier, V Sorge, and R McCasland. Automatic generation of classification theorems for finite algebras. In *Proceedings of IJCAR* , 2004.
4. A Meier and V Sorge. Applying sat solving in classification of finite algebras. *Journal of Automated Reasoning*, 35(1–3):201–235, 2005.
5. V Sorge, A Meier, R McCasland, and S Colton. The automatic construction of isotopy invariants. In *Proceedings of IJCAR*, 2006.
6. H. Pflugfelder. Quasigroups and Loops: Introduction. Heldermann Verlag, 1990.
7. C Weidenbach, U Brahm, T Hillenbrand, E Keen, C Theobald, and D Topic. SPASS version 2.0. In Andrei Voronkov, editor, *Proceedings of CADE–18*, 2002.