Automatic Generation of Classification Theorems for Finite Algebras

Simon Colton[1], Andreas Meier[2⋆], Volker Sorge[3⋆⋆], and Roy McCasland[4⋆⋆⋆]

[1] Department of Computing, Imperial College London, UK,
sgc@doc.ic.ac.uk, http://www.doc.ic.ac.uk/~sgc
[2] DFKI GmbH, Saarbrücken, Germany,
ameier@ags.uni-sb.de, http://www.ags.uni-sb.de/~ameier
[3] School of Computer Science, University of Birmingham, UK,
V.Sorge@cs.bham.ac.uk, http://www.cs.bham.ac.uk/~vxs
[4] School of Informatics, University of Edinburgh, UK
rmccasla@inf.ed.ac.uk, http://www.inf.ed.ac.uk/~rmccasla

Abstract. Classifying finite algebraic structures has been a major motivation behind much research in pure mathematics. Automated techniques have aided in this process, but this has largely been at a quantitative level. In contrast, we present a qualitative approach which produces verified theorems, which classify algebras of a particular type and size into isomorphism classes. We describe both a semi-automated and a fully automated bootstrapping approach to building and verifying classification theorems. In the latter case, we have implemented a procedure which takes the axioms of the algebra and produces a decision tree embedding a fully verified classification theorem. This has been achieved by the integration (and improvement) of a number of automated reasoning techniques: we use the Mace model generator, the HR and C4.5 machine learning systems, the Spass theorem prover, and the Gap computer algebra system to reduce the complexity of the problems given to Spass. We demonstrate the power of this approach by classifying loops, groups, monoids and quasigroups of various sizes.

1 Introduction

As witnessed by the classification of finite simple groups – described as one of the major intellectual achievements of the twentieth century [4] – classifying finite algebraic structures has been a major motivation behind much research in pure mathematics. As discussed further in Sec. 2.2, automated techniques have aided this process, but this has largely been at a quantitative level, e.g., to count the number of groups of a particular order. Classification theorems of a more qualitative nature are often more interesting and more informative, sometimes allowing one to use properties of relatively small structures to help

⋆ The author's work supported by EU IHP grant Calculemus HPRN-CT-2000-00102.
⋆⋆ The author's work was supported by a Marie-Curie Grant from the European Union.
⋆⋆⋆ The author's work was supported by EPSRC MathFIT grant GR/S31099.

classify larger structures. For example, Kronecker's classification of finite Abelian groups [6] states that every Abelian group, G, of size n can be expressed as a direct product of cyclic groups, $G = C_{s_1} \times \cdots \times C_{s_m}$, where $n = s_1.s_2 \ldots s_m$ such that each s_{i+1} divides s_i.

In this paper we look at automating the task of generating and fully verifying qualitative classification theorems for algebraic structures of a given size. As a simple example, our system is given the axioms of group theory and told to find a classification theorem for groups of size 6. It returns the following (paraphrased) result: "all groups of size 6 can be classified up to isomorphism as either Abelian or non-Abelian" [an Abelian group, G, is such that $\forall a, b \in G.\ a \circ b = b \circ a$]. The system generates such results, then proves that they provide valid classifications – as specified in Sec. 2.1 – by showing that each concept is a *classifying property*, i.e., true for all members of exactly *one* isomorphism class.

In our first – semi-automated – approach to generating and verifying classification theorems, as discussed in Sec. 3, the Mace model generator [9] was used to generate representatives of each isomorphism class for the given algebra of given size, then the HR [1] and C4.5 [12] machine learning systems were used to induce a set of classifying properties. To guarantee the validity of the classification we construct appropriate verification problems, which we first simplify with the Gap computer algebra system [3] and then prove with the Spass theorem prover [16]. We found various limitations with this approach, and the lessons we learned informed a second approach. As discussed in Sec. 4, we implemented a fully automated bootstrapping procedure that builds a decision tree which can be used to decide the isomorphism class of a given algebra. The system uses Mace to successively construct non-isomorphic algebras and HR to find properties that discriminate between them. The correctness of the decision tree is guaranteed in each step with Spass after using Gap to simplify the problems.

We used both approaches to generate a number of classification theorems for groups, monoids, quasigroups and loops up to size 6, with the results from the second approach presented in Sec 5. The power of this bootstrapping approach is demonstrated by the generation and verification of a classification theorem which covers the 109 loops of size 6. Not only does our approach highlight the utility of employing multiple reasoning systems for difficult tasks such as classification, our technique is neither restricted to the algebraic domain nor the isomorphism equivalence relation. In Sec. 6, we discuss possible applications of our approach to other domains of pure mathematics, along with other future directions for this work, including distributing the bootstrapping algorithm and producing classification theorems which are generative as well as descriptive.

2 Background

2.1 Classification Problems

We define a general classification problem as follows: let $\mathfrak{A}$ be a finite collection of algebraic structures and let $\sim$ be an equivalence relation on $\mathfrak{A}$. Then $\sim$ induces a partition into equivalence classes $[A_1]_\sim, [A_2]_\sim, \ldots, [A_n]_\sim$, where $A_i \in \mathfrak{A}$ for $i = 1, \ldots, n$.

Let P be a property which is invariant with respect to $\sim$. Then P acts as a *discriminant* for any two structures A and B in $\mathfrak{A}$, in the sense that if $P(A)$ and $\neg P(B)$, then $A \not\sim B$. If, in addition, P holds for every element of an equivalence class $[A_i]_\sim$, but does not hold for any element in $\mathfrak{A} \setminus [A_i]_\sim$, then we call P a *classifying property* for $[A_i]_\sim$. A full set of classifying properties – with one property for each equivalence class – comprises a classifying theorem stating that each element of $\mathfrak{A}$ exhibits exactly one of the classifying properties. The classification problem is therefore to find a full set of classifying properties.

At present we consider the isomorphism equivalence relation $\cong$, and the algebraic structures quasigroups, loops, groups and monoids. For these structures, we only need the following four axioms (letting A be a set and $\circ$ be a closed operation on the elements of A):

1. **Associativity:** A is associative with respect to $\circ$.
2. **Divisors:** For every two elements $a, b \in A$ there exist two corresponding divisors $x, y \in A$ such that $a \circ x = b$ and $y \circ a = b$ holds.
3. **Unit element:** There exists a unit element e in A with respect to $\circ$.
4. **Inverse:** Given a unique unit element e, each element x has an inverse x^{-1} such that $x \circ x^{-1} = x^{-1} \circ x = e$.

We call $(A, \circ)$ a *quasigroup* if only axiom 2 holds, a *monoid* if axioms 1 and 3 hold, a *loop* if axioms 2 and 3 hold and a *group* if axioms 1, 3 and 4 hold. Working with relatively simple algebras keeps the necessary axiomatic overhead down, but can lead to large numbers of structures even for small sizes, e.g., there are 109 non-isomorphic loops of size 6.

As a concrete example, consider quasigroups of size 3. There are 12 quasigroups of this size, but only 5 different isomorphism classes. Presented in terms of their Cayley tables, the following three quasigroups of order 3 are pairwise non-isomorphic.

Q_1	a	b	c
a	b	a	c
b	a	c	b
c	c	b	a

Q_2	a	b	c
a	a	c	b
b	c	b	a
c	b	a	c

Q_3	a	b	c
a	a	b	c
b	b	c	a
c	c	a	b

Because if we observe the diagonals of the Cayley tables for Q_1 and Q_2, we find that Q_2 has idempotent elements (i.e., such that $x \circ x = x$), but Q_1 does not. Hence the property $\exists x. x \circ x = x$ is a discriminant for Q_1 and Q_2. However, this property is not enough to distinguish Q_2 from Q_3 (i.e., this property is not a classifying property for $[Q_2]_\cong$), as Q_3 also contains idempotent elements. Since all elements of Q_2 are idempotent, we can strengthen the property to $\forall x. x \circ x = x$, which is sufficient to discriminate Q_2 from both Q_1 and Q_3. As we will see later, this is actually a classifying property for the equivalence class represented by Q_2.

2.2 Previous Work

When constructing classification theorems for algebraic structures, the first question to answer is how many algebras there are for a given size. Automated tech-

niques such as constraint solving and the Davis-Putnam method have been used extensively to determine the number of algebras of a given type and size, and this has answered many open questions. In particular, the Finder system has been used to solve many quasigroup existence problems [14]. Also, representatives of every isotopy and isomorphism class for quasigroups and loops have been generated up to at least order 9 [10]. In addition to classifying structures within an algebraic axiomatisation, automated theorem proving has been used to find new axiomatisations for algebras, thus enabling better intra-classification of algebras. In particular, new axiomatic representations of algebras such as groups have been found [5, 8].

As described in [1], integration of automated reasoning systems has always been a major aspect of the HR project. Moreover, we have shown in previous work that the HR system can be used to support proof planning, and that this has some promise for classification tasks [11]. However, to our knowledge, there have been no attempts to produce and verify full classification theorems for particular algebras of a given size from the axioms alone. As described in Sec. 3 and Sec. 4, our approach has been to integrate various reasoning systems, as we believe it is not possible for a single system to solve this problem. In particular, we have employed the following systems:

Spass is a resolution-based automated theorem prover for first-order logic with equality [16]. Spass combines a superposition calculus with specific inference/reduction rules for sorts and a splitting rule for case analysis.

MACE-4 is a model generator that searches for finite models of first-order formulae [9]. For a given domain size, all instances of the formula over the domain are constructed and a decision procedure searches for satisfiable instances of the formula. The distribution package contains several auxiliary programs, including the *isofilter* program, which detects and removes isomorphic models, returning a set of pairwise non-isomorphic models.

HR is a machine learning program which performs automated theory formation by building new concepts from old ones [1] using a set of production rules. It uses the examples of the concepts to empirically form conjectures relating their definitions [2] and employs third party theorem proving and model generation software to prove/disprove the conjectures.

C4.5 is a state of the art decision tree learning system which has been used with much success for many predictive induction tasks [12].

Gap is a computer algebra system with special application to algebraic domains [3]. We used Gap as a toolbox to implement various computer algebra algorithms involving generators and factorisations of algebras.

3 Approach One: Semi-automated

3.1 Stage 1: Generating Non-isomorphic Algebras

The goal of this stage was to produce a covering set of pairwise non-isomorphic algebras of the given type and size, i.e., a single representative of each isomorphism class. To do this, we used Mace in two parts: by constructing all structures

with the given properties and cardinality, then applying the isofilter tool to remove isomorphic structures. We also used lists of loops up to size 7 and lists of quasigroups up to size 5 kindly provided by Wendy Myrvold.

3.2 Stage 2: Generating Classifying Concepts

Given a set of pairwise non-isomorphic algebras, $A_1, \ldots, A_n$ such as those generated in stage 1, the problem in stage 2 was to find a set of boolean properties $P_1, \ldots, P_n$ such that P_i is a classifying property for $[A_i]_{\cong}$, for all i. To do this, we used only HR's concept formation abilities. Starting with some background concepts – in this case, those derived from the axioms of the algebras being considered, e.g., multiplication, inverse, identity, etc., – HR builds new concepts from old ones using a set of production rules. For this application, we used the compose, exists, match and negate production rules. To see how these work, consider starting in group theory with the concept of triples of elements $[a, b, c]$ related via the multiplication concept, i.e., $a \circ b = c$. In its search for concepts, HR might use the compose rule (which conjoins clauses in definitions) to compose this concept with itself, producing the concept of triples of elements related via commutativity: $[a, b, c]$ such that $a \circ b = c \wedge b \circ a = c$. Later, HR might use the negate production rule (which negates clauses in definitions) to produce this concept: $[a, b, c]$ such that $a \circ b = c \wedge b \circ a \neq c$. It then might use the exists production rule (which introduces existential quantification) to produce the concept of non-Abelianess, i.e., groups for which $\exists a, b, c . a \circ b = c \wedge b \circ a \neq c$. Finally, it might use the negate rule again to produce the concept of Abelianess: groups for which $\nexists a, b, c . a \circ b = c \wedge b \circ a \neq c$. In this way, we can see how HR can solve the classification problem for groups of size 6, as mentioned in Sec. 1.

We started by simply using HR to exhaustively generate concepts until it had produced boolean properties $P_1, \ldots, P_n$ as required above. This scheme worked for some simple classification problems, but it did not scale up, and various improvements were made. Firstly, we used a heuristic search whereby the match, exists and negate production rules were used greedily before the compose rule was used. This encourages the generation of the kind of boolean properties we require, and greatly increased the efficiency of HR's search. For instance, using a breadth first search, HR takes over an hour to solve the classification of size 3 quasigroups. However, with the greedy heuristic search, this took only 6 seconds (on a Pentium 4 2GHz machine), producing the following theorem:

Quasigroups of size 3 have exactly one of the following properties:
(i) $\nexists a . a \circ a = a$ (ii) $\forall a . a \circ a = a$ (iii) $\exists a . \nexists b, c . b \circ c = a \wedge a \circ b = c$
(iv) $\exists a . \nexists b, c . b \circ a = c \wedge c \circ b = a$ (v) $\exists a . \nexists b, c . a \circ b = c \wedge c \circ a = b$.

Using this simple scheme, HR was able to solve classification problems for groups of size 6 and 8, loops of size 4 and 5, quasigroups of size 3, qg4-quasigroups of size 5 and qg5-quasigroups of size 7 [note that qg4-quasigroups are quasigroups such that $\forall a, b . (b \circ a) \circ (a \circ b) = a$ and qg5-quasigroups are such that $\forall a, b . (((b \circ a) \circ b) \circ b) = a$]. As an interesting example of the kind of result produced in this way, in a session working with the five isomorphism classes for groups of size

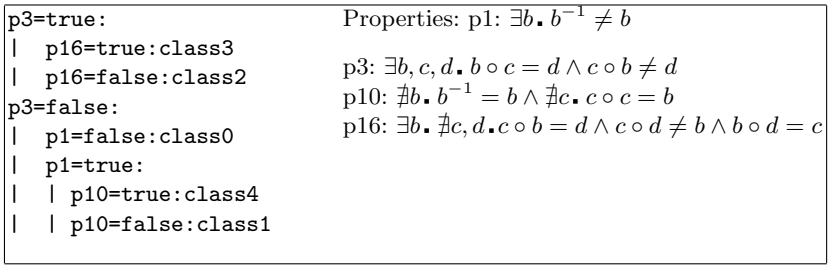

Fig. 1. Decision tree produced by HR/C4.5 for groups of size 8

8, we gave HR the additional concept of commutators (elements which can be expressed as $a \circ b \circ a^{-1} \circ b^{-1}$ for some a and b), and HR produced four classifying concepts, which enabled us to form this (paraphrased) classifying theorem:

Groups of order 8 can be classified by their self-inverse elements ($x^{-1} = x$): they will either have (i) all self inverse elements (ii) an element which squares to give a non-self inverse element (iii) no self-inverse elements which aren't also commutators (iv) a self inverse element which can be expressed as the product of two non-commutative elements or (v) none of these properties.

As a second improvement, we used HR differently: we asked it to form a theory for a given amount of time (up to an hour), then asked it to output all the boolean properties it had produced, regardless of whether they were classifying properties or not. The output was produced in Prolog format, and we used Sicstus Prolog to search for conjunctions of the boolean properties which were true of single algebras. This method helped us scale up further: we were able to find five classifiers for groups of size 8 much more efficiently than with HR alone. More interestingly, we solved the classification problem of quasigroups of size 4 (for which there are 35 isomorphism classes). Using HR for 20 minutes, followed by a Prolog search lasting 5 minutes, the classification theorem produced had 9 classifying properties which were single boolean properties from HR, 25 classifying properties which were conjunctions of two boolean properties and 1 classifying property which was a conjunction of three boolean properties.

As an alternative to using Prolog, we experimented using C4.5: by giving it the boolean properties generated by HR and making the isomorphism classification the one for C4.5 to learn, we were able to produce decision trees such as the one in Fig. 1 (for groups of size 8, presented in the format produced by C4.5). We see that conjoining the nodes of the tree from root to leaf for each leaf provides a set of classifying properties, hence a classification theorem can be derived from the tree. Using C4.5 was problematic, however because (a) due to statistical constraints, we had to multiply our data, e.g., give 10 copies of each algebra, before the algorithm would learn a tree and (b) due to a heuristic search, C4.5 would sometimes learn a tree which was less than 100% accurate, hence could not be used as a theorem. However, we believe that with more experimentation, we will be able to overcome these difficulties.

Using one or other of the above schemes, we found classification theorems for each of the algebra/sizes we tried, with one exception: the classification problem for loops of size 6, with 109 isomorphism classes. For this problem, using the Prolog extension, we found classifying properties for only around half the isomorphism classes, and using the decision tree extension, C4.5 produced a tree, but it was not 100% accurate and didn't cover all the classes.

3.3 Stage 3: Verifying the Classification

None of HR, Mace or C4.5 have been formally verified to work correctly. Hence, it was necessary to use an automated theorem prover to prove that the classification theorems were correct. HR and C4.5 provide classification results in the form of a set of pairs $(A_1, P_1), \ldots, (A_n, P_n)$ of structures A_i and discriminant properties P_i. We call these *classification pairs* since P_i characterises an isomorphism class of algebras with axiomatic properties $\mathcal{P}$ and cardinality c, and A_i is a representant for this isomorphism class. To verify a classification result, we prove the following theorems with Spass:

Discriminant Theorems: Each property P_i is a discriminant, i.e., if P_i holds for one algebra but does not hold for another algebra, then the two algebras are not isomorphic. This also guarantees that P_i is an invariant under isomorphism, since this property is logically equivalent.
Representant Theorems: For each pair (A_i, P_i), A_i satisfies P_i and $\mathcal{P}$.
Non-Isomorphic Theorems: For two pairs (A_i, P_i) and (A_j, P_j) the representants A_i and A_j are not isomorphic. This verifies the results of Mace and guarantees against the construction of too many isomorphism classes.
Isomorphism-Class Theorems: For each pair (A_i, P_i), all algebras of cardinality c that satisfy P_i and $\mathcal{P}$ are isomorphic.
The Covering Theorem: For each algebra of cardinality c that satisfies $\mathcal{P}$, $P_1 \vee \ldots \vee P_n$ holds. This guarantees that all isomorphism classes are found.

Properties P_i are invariants (i.e., discriminants) for arbitrary algebras. Thus, the discriminant theorems do not depend on the properties $\mathcal{P}$ and the cardinality c. To encode the discriminant theorems for proof by Spass, we employ two sort predicates s_1 and s_2 and two binary operations $\circ_1$ and $\circ_2$ to model two algebras, which are closed with respect to s_1 and s_2, respectively. The required discriminant properties are encoded with sorted quantifiers. Both kinds of theorems require the encoding of an arbitrary isomorphism h between the two algebras.

The other theorems depend on the properties $\mathcal{P}$ and P_i as well as the cardinality c (e.g., a property P_i (typically) specifies an isomorphism class only for a particular $\mathcal{P}$ and c). For the covering and isomorphism-class theorems, we have to model arbitrary algebras of cardinality c. In our experiments, we discovered that Spass performed best on these problems when transforming the first-order encoding into a propositional logic encoding without quantifiers (we also tested an encoding with sorts and sorted quantifiers). We first encode the fact that there are c different elements $e_1, \ldots, e_c$. Then, the quantifiers of all used properties are expanded with respect to $e_1, \ldots, e_c$: a universal quantification $\forall x. Q[x]$

results in the conjunction $Q[e_1] \wedge \ldots \wedge Q[e_c]$, whereas an existential quantification $\exists x. Q[x]$ results in the disjunction $Q[e_1] \vee \ldots \vee Q[e_c]$.

For the isomorphism-class theorems, we have to prove the existence of an isomorphism between two arbitrary algebras satisfying P_i and $\mathcal{P}$, which is generally a much harder task than constructing an isomorphism for two concrete structures. As we deal only with finite algebras of given cardinality c, we can enumerate the $c!$ possible bijective functions between two algebras of cardinality c and explicitly axiomatise them as pointwise defined functions. Then, the conclusion of an isomorphism-class theorem is that (at least) one of the bijective functions is a homomorphism, i.e., $homo(h_1) \vee \ldots \vee homo(h_{c!})$. The isomorphism-class theorems turned out to be the most complex problems for Spass. In particular, their complexity heavily depends on the cardinality, c, since there are $c!$ potential bijective functions. In order to reduce the complexity of these theorems, we consider sets of generators and factorisations to decrease the number of potential isomorphism mappings. In this context, a structure $(A, \circ)$ is generated by a set of elements $\{a_1, \ldots, a_m\} \subseteq A$ if every element of A can be expressed as a combination – usually called a factorisation or word – of the a_i under the operation $\circ$. For example, quasigroup Q_3 from Sec. 2.1 is generated by element $b \in A$, as both $c = b \circ b$ and $a = b \circ (b \circ b)$ can be expressed as factorisations in b.

Note that generators and factorisations are invariants of isomorphisms. That is, if a structure $(A, \circ)$ is generated by $\{a_1, \ldots, a_m\} \subseteq A$ and h is an isomorphism mapping $(A, \circ)$ to a structure $(A', \circ')$, then $\{h(a_1), \ldots, h(a_m)\} \subseteq A'$ is a generating set for $(A', \circ')$. Moreover, given a factorisation, in terms of the a_i's, for an element $a \in A$, then one obtains the factorisation for $h(a)$ by simply replacing each of the a_i's with $h(a_i)$, respectively.

To compute generators and factorisations we employed a computer algebra algorithm, which we encoded in the Gap system. For a given structure, this algorithm constructs a minimal set of generators together with a set of factorisations expressing each element of the structure in terms of the generators. The set of generators is constructed by successively combining elements with the longest traces until all elements of the set can be generated. If necessary, the set of generators is then reduced if any of its elements is redundant. Note that this approach works for all types of algebraic structures, regardless of the number of operations, and ensures that the set of generators is minimal in the sense that none of its subsets generates the full set.

If a classification pair (A_i, P_i) characterises an isomorphism class and if our algorithm computes generators $\{a_1, \ldots, a_m\} \subseteq A_i$ and factorisations for A_i, then all algebras with property P_i (and cardinality c and properties $\mathcal{P}$) have a generating set with m elements and the factorisations of the elements of A_i. We prove this as additional theorem which we call the *Gensys-Verification Theorem*. Having proved this, we can express the isomorphism-class theorem for the algebras using the generators and factorisations. This reduces the number of functions which are candidates for isomorphisms. Instead of $c!$, there are only $\frac{c!}{(c-m)!}$ possible mappings, since only the m generators have to be mapped ex-

plicitly. This is because each isomorphism is uniquely determined by the images of these generators.

We ran Spass on a Linux PC with four 2GHz Xeon processors and 4GB RAM and proved HR's groups6, loops4, loops5, quasigroups3 and qg4-quasigroups5 results. For the quasigroup4 classification problem, Spass failed due to internal problems while proving the covering theorem. For the qg5-quasigroups7 and the groups8 results, Spass was not able to prove all the gensys-verification theorems within a two day time limit. For the discriminant theorems, Spass typically required less than a second. However, there were a few examples which needed considerably more time (e.g., one proved theorem in the quasigroup4 classification problem was to show that $\forall x. \exists y. (x \circ y) \circ (x \circ y) = y$ is a discriminant; Spass needed 1h 6m to prove this theorem). The performance of Spass on representant and non-isomorphic theorems depended slightly on the cardinality of the examined algebras. Spass needed less than a second to prove the theorems for the smaller algebras, but took several seconds to prove the theorems for the larger algebras.

The complexity of covering theorems depends not only on the cardinality of the examined algebras, but also on the number of isomorphism classes and the classifying properties. Unfortunately, Spass failed on the covering theorem resulting from the 35 quasigroups4 isomorphism classes, but it proved the groups8 covering theorem in about 4 seconds (this theorem is particularly simple since it has the form $Q \vee \neg Q$ where $Q = P_1 \wedge P_2 \wedge P_3 \wedge P_4$), and it took about 4 minutes to prove the loops5 covering theorem. In addition, isomorphism-class and gensys-verification theorems depend heavily on the cardinality of the examined algebras, as we shall discuss further in Sec. 5.

4 Approach Two: Bootstrapping

After experimenting with various schemes in approach 1, we identified some limitations, which enabled us to specify the following requirements for an algorithm in our second approach:

- The process should be entirely automatic and bootstrapping, able to produce verified classification theorems starting from the axioms alone, with no human intervention.
- The process should call HR to find classifying concepts for small numbers of isomorphism classes, as HR struggled to solve larger classification problems.
- The process should not require the production of *every* algebraic structure satisfying the axioms. This is because, when using Mace to generate all structures and reduce them using its isomorphism filter, we found that the intermediate files produced could often be unmanageably large (up to 4GB).
- The process should generate the classification theorem as a decision tree. We found that decision trees often involved fewer concept definitions and enabled easier classification of algebras.

Input: Cardinality c and basic properties $\mathcal{P}$ of the algebraic structures
Output: Decision tree $(r, \mathcal{V}, \mathcal{E})$

1. **Initialise** $\mathcal{V} := \{r\}$, $\mathcal{E} := \emptyset$.
2. Let $\mathcal{S}$ be the set of unprocessed nodes of the tree, initially $\mathcal{S} := \{r\}$.
3. **While** $\mathcal{S} \neq \emptyset$ do
   3.1. Pick $v \in \mathcal{S}$ with $l(v)=P_v$ (i.e. the label of v specifies the properties P_v)
   3.2. **If** there exists a model m that satisfies $\mathcal{P} \cup P_v$ then:
      3.2.1. **If** there exists $m' \ncong m$ satisfying $\mathcal{P} \cup P_v$ then:
         3.2.1.1. Construct discriminants P_1 and P_2 for m and m', respectively.
         3.2.1.2. **If** $P_1=\neg P_2$ then create two child vertices v_1, v_2 with edges e_1, e_2 such that $l(e_1)=P_1, l(e_2)=P_2$. $\mathcal{S} := \mathcal{S} \cup \{v_1, v_2\}$.
         3.2.1.3. **Else** create four child vertices v_1, v_2, v_3, v_4 and edges $e_1, \ldots, e_4$ such that $l(e_1)=P_1 \wedge P_2$, $l(e_3)=P_1 \wedge \neg P_2$, $l(e_2)=\neg P_1 \wedge P_2$, and $l(e_4) = \neg P_1 \wedge \neg P_2$. $\mathcal{S} := \mathcal{S} \cup \{v_1, v_2, v_3, v_4\}$.
      3.2.2. **Else** all structures of cardinality c with $\mathcal{P} \cup P_v$ are isomorphic: Mark v as a leaf representing an isomorphism class.
   3.3. **Else** no structures of cardinality c with $\mathcal{P} \cup P_v$ exist: Mark v as empty leaf.
   3.4. $\mathcal{S} := \mathcal{S} \setminus \{v\}$.
4. **Return** $(r, \mathcal{V}, \mathcal{E})$.

Fig. 2. The bootstrapping algorithm used in approach 2.

The algorithm portrayed in Fig. 2 satisfies these criteria. This constructs a decision tree by successively generating non-isomorphic algebraic structures and associated discriminants until the maximal number of isomorphism classes is reached. The result serves as a qualitative classification for the specified algebras up to isomorphism for a given cardinality. The algorithm takes the cardinality c and axiomatic properties $\mathcal{P}$ of the algebraic structures to be considered as input. It returns a decision tree $(r, \mathcal{V}, \mathcal{E})$ for isomorphism classes of the algebraic structure, where $\mathcal{V}$ is a set of vertices, $\mathcal{E}$ a set of edges, and $r \in \mathcal{V}$ is the root of the tree. Each edge $e \in \mathcal{E}$ is labelled with an algebraic property and each vertex $v \in \mathcal{V}$ is labelled with the conjunction of properties on the path from r to v.

In principle the algorithm is complete, i.e., for given cardinality c and axiomatic properties $\mathcal{P}$ it provides a classification theorem after a finite number of steps. In practice, however, the success of the algorithm depends on whether all used systems actually provide the answers they are supposed to deliver (see Sec. 5 for the discussion of the practical limitations of the algorithm).

To illustrate the algorithm, we use the example of the decision tree constructed to classify quasigroups of order 3, given in Fig. 3 (here, the vertices of the tree are enumerated rather than assigned their actual labels, to preserve space.)

Initially, the decision tree consists only of the root node. The single nodes of the tree are expanded by first generating an algebraic structure satisfying the properties specified by the node (step 3.2). For example, for the root node 1 in Fig. 3, an arbitrary quasigroup of order 3 is constructed; for node 3, however, a

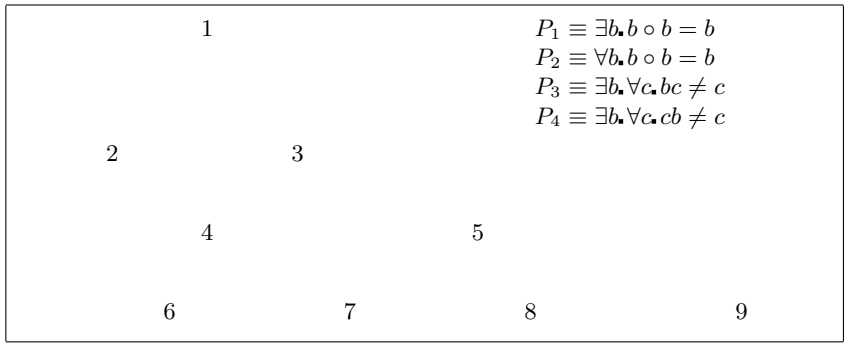

Fig. 3. Decision tree for the classification problem of order 3 quasigroups.

$\neg P_3 \wedge P_4$

quasigroup that additionally satisfies the property $\exists b. b \circ b = b$ is required. After such a *representant* is generated by Mace, we check with Spass that it actually satisfies the required properties. The first representant generated in our example was quasigroup A_2 given in Fig. 4. If no representant can be generated, we must prove that there exists no algebraic system of cardinality c satisfying both the original axioms $\mathcal{P}$ and the additional properties generated up to this point (step 3.3). This proof is again done by Spass. An example of this case is node 9, i.e., no quasigroup of size 3 can be constructed which satisfies $P_1 \wedge \neg P_2 \wedge \neg P_3 \wedge \neg P_4$.

When Mace does produce a model, we have two cases to consider: there exists a non-isomorphic structure exhibiting the same properties considered so far (step 3.2.1) or the property represented by the node constrains the structures to a single isomorphism class (step 3.2.2). In the latter case, we employ Spass to prove the appropriate isomorphism-class theorem (using Gap as in approach 1 to re-write the theorems in terms of generators and factorisations). For instance, in node 2 we show that all quasigroups of order 3 without idempotent elements belong to one isomorphism class. In the former case, we employ Mace to generate a structure non-isomorphic to the original one. The two structures are then passed to HR to compute two discriminants (step 3.2.1.1). Again, Spass does the necessary verification of the non-isomorphic theorem and the discriminant theorems. In the example, the quasigroup A_4 given in Fig. 4 was constructed as a non-isomorphic counterpart to A_2. Given those two quasigroups, HR came up with discriminants P_1 and $\neg P_1$ as stated in Fig. 3. Depending on the nature of the discriminants, either two or four child nodes are constructed (steps 3.2.1.2 & 3.2.1.3). The case of two child nodes can be observed in the expansion of nodes 1 and 3 in Fig. 3, whereas the expansion of node 5 leads to four children. The

A_2	a	b	c
a	b	a	c
b	a	c	b
c	c	b	a

A_4	a	b	c
a	a	c	b
b	c	b	a
c	b	a	c

A_6	a	b	c
a	a	b	c
b	b	c	a
c	c	a	b

A_7	a	b	c
a	a	b	c
b	c	a	b
c	b	c	a

A_8	a	b	c
a	a	c	b
b	b	a	c
c	c	b	a

Fig. 4. Isomorphism class representants for quasigroups of order 3.

advantage of the former case is that when expanding the two child nodes, the models for step 3.2 can be reused and do not have to be produced with Mace, whereas in the second case, Mace has to be called for two of the nodes.

Once the decision tree is fully expanded, the discriminating properties of all the isomorphism classes give the final classification theorem. In our example, this corresponds to a disjunction of the labels of the doubly-circled leaf nodes. Although we omit a formal proof, both the construction of the decision tree as well as the fact that we work in classical logic (i.e. *tertium non datur* holds) guarantee that the final decision tree determines all possible isomorphism classes and a full classification for the algebraic systems specified by the input parameters. Importantly, the correctness of our implementation depends only on the correctness of Spass.

The branching factor of up to four of the tree is influenced by two design decisions for our algorithm: On the one hand, we could have kept the decision tree binary by always branching with respect to one discriminant and its negation. However, this would have meant losing some of the more intuitive discriminants generated by HR, as well as discarding previously computed results and thus increasing the number of calls to HR. Alternatively, we could have given HR more than two non-isomorphic structures to compute discriminants for, thereby reducing the number of calls to HR and increasing the branching factor of the tree. We decided against this, because this increases the risk that HR would not come up with an answer, or produce lengthier, more complex discriminants, which are usually more difficult to verify with Spass.

Both the size of the decision tree and the number of calls to Spass and Mace can become fairly large, even when considering a relatively small number of structures. However, the algorithm offers some potential for parallelism and distribution. We currently exploit this by parallelising steps 3.2.1 and 3.2.2, i.e. generating a non-isomorphic structure with Mace and proving the isomorphism-class theorem with Spass. This actually increases efficiency, since both computations are generally very expensive and take a long time. We gain another small speed up by constructing the proofs for step 3.2.1.1 in parallel with Spass. We highlight more potential for parallelism in Sec. 6.

5 Results from the Bootstrapping Approach

Given the more ad-hoc, semi-automated nature of approach 1 when compared to approach 2, and given that approach 2 was found to be more effective, while we have supplied some illustrative results for approach 1 in Sec. 3, we concentrate here on the testing we undertook for approach 2. We tested the hypothesis that the bootstrapping system can generate and verify full classification theorems for loops, groups, quasigroups and monoids up to size 6. Hence we experimented by using the system to generate classification theorems, and Table 1 summarises the results of these experiments. For each algebra, the table describes the decision tree constructed to classify it, in terms of the number of nodes, the number of identified isomorphism classes, and the maximal depth of the tree. Moreover, it

Algebra	Nodes	IsoClasses	Max-Depth	Isoclass Proof	Gensys-Verification Proof
Monoids3	13	7	5	< 1s	< 1s
Quasigroups3	9	5	4	< 1s	< 1s
Quasigroups4	71	35	9	17s	50s
Loops4	3	2	2	2s	< 1s
Loops5	11	6	5	21s	45s
Loops6	233	109	17	3m4s	668m17s
Groups4	3	2	2	< 1s	< 1s
Groups6	3	2	2	1m40s	6m10s

Table 1. Results of the experiments with the bootstrapping approach.

provides the mean times of Spass runs to prove the necessary isomorphism-class and gensys-verification theorems, i.e., the most complex proof problems.

We see that the proof time taken by Spass significantly increases with the cardinality of the algebra as well as with the depth of the trees (the higher the depth, the more properties associated with the nodes). In particular, these statistics suggest that cardinality 7 is the borderline for the discussed techniques, since we cannot hope to prove, in particular, gensys-verification theorems beyond cardinality 8.[1] However, we believe it is a significant achievement to produce a classification theorem for the 109 isomorphism classes of loops of size 6. Moreover, we currently employ Spass in auto-mode, and it might be possible to push the solvability horizon with settings tailored to our problems.

The time Mace needs to construct a model for a node in a tree depends on both the cardinality of the structures and the depth of the node in the tree. The deeper the node in the tree, the more properties the model has to satisfy. For instance, for the initial nodes of Loops6, Mace needed less than a second to construct a model. For the deep nodes it needed up to 15 seconds. The time Mace needs to construct a non-isomorphic structure is typically considerably longer (up to 4 minutes for Loops6). This is not surprising, as the encoding of these Mace problems comprises many additional symbols for the homomorphisms. For nodes that correspond to isomorphism classes, no non-isomorphic structures exist, and for these cases, Mace had to traverse the entire search space, which could take up to an hour for Loops6.

In the experiments summarised above, the bootstrapping algorithm always computed only two non-isomorphic structures, which it passed to HR, and HR succeeded in finding suitable discriminants in every case. Moreover, in no cases were Mace or HR shown to have performed incorrect calculations. We also experimented for Loops6 with settings that passed 3 and 4 non-isomorphic structures together to HR. In these experiments HR often took considerably longer and sometimes failed to provide discriminants. Moreover, when successful, it created discriminants that were considerably more complex as they involved more quan-

[1] HR succeeded to classify groups of order 8. In approach 1, when verifying the classification of HR, Spass was able to prove all 5 isomorphism-class theorems (mean time: around 7 hours) but succeeded to prove only one gensys-verification theorem (in about 19 hours). We interrupted the other runs of Spass after 3 days.

tifiers and sub-formulas. These in turn made the verification problems much more difficult for Spass.

6 Conclusions and Further Work

We have presented a novel approach to constructing classification theorems in pure mathematics, which has produced novel and interesting results of a qualitative nature. The classification theorems produced have often contained classically interesting results such as commutativity and idempotency, and we believe, especially for the larger classification problems, that no such theorems have ever been produced. Our bootstrapping algorithm successfully exploits the strengths of diverse reasoning techniques including deduction, inductive learning, model generation and symbolic manipulation, while avoiding many weaknesses we identified in earlier approaches, which we have also presented. The collaboration of the various systems produces results that clearly cannot be achieved by any single system, and the incorporation of external systems offers improved flexibility, as we can profit from any advances of the individual technologies.

Thus far we have dealt only with isomorphism classes of relatively simple algebraic structures. However, our approach is adaptable to more complicated algebraic domains and indeed Spass, Mace and HR have all been demonstrated to work in algebras over two operators like rings. In addition, we also want to experiment with different types of equivalence relations, which can lead to more insights about the structures under consideration. For instance, quasigroups and loops can also be grouped into isotopy classes, and modules can be distinguished with respect to their uniform dimensions. Moreover, classification is not only interesting in algebraic domains, and our approach could also be applied to other domains such as analysis, differential geometry or number theory.

In addition to scaling up in terms of the complexity of the domains looked at, we also hope to produce classification theorems for larger sizes. The bootstrapping approach provides much potential for parallelisation, some of which we already exploit in our current implementation. In addition, the decision tree offers potential for distribution since new threads can be spawned for already existing vertices by applying the algorithm to the original axioms extended by the properties a vertex is labelled with. Nevertheless, the results of our experiments suggest that cardinality 8 is the borderline of our current approach. A general first-order theorem prover like Spass seems unable to prove isomorphism-class and gensys-verification problems with larger cardinalities. There are two possible solutions to further push the solvability horizon. Firstly, we could try to involve further symbolic computations to simplify the problems at hand. Secondly, special theorem proving techniques for finite algebras could be developed and employed.

In mathematics, classifying theorems are often generative, e.g., Kronecker showed that, by constructing certain direct products of cyclic groups, it is possible to generate every Abelian group. To produce such generative theorems, it may be necessary to more systematically construct discriminants in order to

gain comparable properties for structures of diverse cardinality, and to work with more complicated concepts such as maps between algebraic structures and products of algebras, such as the direct product. We believe that as systems such as the one we have presented here become more sophisticated, they may be of use to pure mathematicians. For instance, currently only Abelian quasigroups are classified [13], whereas we have classified all quasigroups, albeit only up to a certain size. We intend to analyse the classification theorems produced by the system in order to identify some concepts which may form a part of a general classification theorem. In particular, we intend to work with algebras associated with Zariski spaces [7], a relatively new domain of pure mathematics which we hope to explore via automated means.

Acknowledgements

We would like to thank Wendy Myrvold and colleagues for providing data on isomorphism classes for loops and quasigroups, Bill McCune for expert advice about Mace, Thomas Hillenbrand for providing us with an improved version of Spass and helping us to encode our proof problems, and Geoff Sutcliffe for helping us to determine that Spass was best suited for our task.

References

1. S Colton. *Automated Theory Formation in Pure Mathematics.* Springer, 2002.
2. S Colton. The HR program for theorem generation. In Voronkov [15].
3. The GAP Group. *GAP – Groups, Algorithms, and Programming, Version 4.3*, 2002. http://www.gap-system.org.
4. J Humphreys. *A Course in Group Theory.* Oxford University Press, 1996.
5. K Kunen. Single Axioms for Groups. *J. of Autom. Reasoning*, 9(3):291–308, 1992.
6. L Kronecker. Auseinandersetzung einiger Eigenschaften der Klassenanzahl idealer komplexer Zahlen. *Monatsbericht der Berliner Akademie*, pages 881–889, 1870.
7. R McCasland, M Moore, and P Smith. An introduction to Zariski spaces over Zariski topologies. *Rocky Mountain Journal of Mathematics*, 28:1357–1369, 1998.
8. W McCune. Single axioms for groups and Abelian groups with various operations. *J. of Autom. Reasoning*, 10(1):1–13, 1993.
9. W McCune. *Mace4 Reference Manual and Guide.* Argonne National Laboratory, 2003. ANL/MCS-TM-264.
10. B McKay, A Meinert, and W Myrvold. Counting small latin squares. European Women in Mathematics Int. Workshop on Groups and Graphs, pages 67–72, 2002.
11. A Meier, V Sorge, and S Colton. Employing theory formation to guide proof planning. In *Proc. of Calculemus-2002*, *LNAI* 2385, pages 275–289. Springer, 2002.
12. R Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, 1993.
13. J. Schwenk. A classification of abelian quasigroups. *Rendiconti di Matematica, Serie VII*, 15:161–172, 1995.
14. J Slaney, M Fujita, and M Stickel. Automated reasoning and exhaustive search: Quasigroup existence problems. *Comp &Math with Applications*, 29:115–132, 1995.
15. A Voronkov, editor. *Proc. of CADE–18*, *LNAI* 2392. Springer, 2002.
16. C Weidenbach, U Brahm, T Hillenbrand, E Keen, C Theobald, and D Topic. SPASS version 2.0. In Voronkov [15], pages 275–279.