Classification results in quasigroup and loop theory via a combination of automated reasoning tools

Volker Sorge, Simon Colton, Roy McCasland, Andreas Meier

Abstract. We present some novel classification results in quasigroup and loop theory. For quasigroups up to size 5 and loops up to size 7, we describe a unique property which determines the isomorphism (and in the case of loops, the isotopism) class for any example. These invariant properties were generated using a variety of automated techniques — including machine learning and computer algebra — which we present here. Moreover, each result has been automatically verified, again using a variety of techniques — including automated theorem proving, computer algebra and satisfiability solving — and we describe our bootstrapping approach to the generation and verification of these classification results.

Keywords: quasigroups, loops, classification, automated reasoning

Classification: 20N05

1. Introduction

Given an equivalence class such as isomorphism or isotopism, in the process of deriving a classification of an algebraic domain such as loops and quasigroups, a natural first step is to *count* the number of equivalence classes for a given size. A natural next step is to *describe* the equivalence classes of a given size in terms of a property shared by all members of a class and by no members of another class. A (possibly infinite) full set of such invariant properties and a mapping from each size onto a subset of these properties constitutes a classification theorem, e.g., in Kronecker's classification of Abelian groups [10] the invariant properties are described in terms of the cross-product decomposition of the group. Automated techniques have been used to good effect for the counting step, and numerous existence problems have been solved in this way, e.g., [19]. We present here an approach to automating the second step, namely to generate classification theorems for particular sizes of quasigroups and loops, which describe the isomorphism/isotopism class structure for each size.

There are two important aspects to this approach. Firstly, the generation of invariant properties is key to the production of the classification theorems. Note that we use the phrase *discriminating* properties interchangeably with *invariant* properties, depending on the context. We have used four different methods to produce these invariants: each method is given a pair of non-equivalent algebraic structures and is asked to determine a property that only one of them has. This is a description of a machine learning problem [14]. Hence, our first approach — as described in Section 3 — used a machine learning system, and derived first-order properties involving only the multiplication symbol and equality. This method was sufficient to produce isomorphism results for loops up to size 6 and quasigroups up to size 5. Looking towards eventually determining classifying properties shared by classes of different sizes (which usually describe *families* of algebraic structures, such as dihedral groups in group theory, etc.), we enhanced the approach to be able to count elements of a particular type. We found that this approach produced simpler classification theorems with more homogeneity across different orders, as described in Section 4.

Turning to isotopism as the equivalence relation for loops, we found that the machine learning approach did not produce isotopic invariants. Instead, we used results from [7] to derive a method for generating equational invariants for loops, as described in Section 5. This too had limitations, so we introduced new methods for using sub-blocks of loops to produce invariants, as described in Section 6. Using a combination of equational and sub-block invariants, we were able to produce isotopism classification results for loops up to size 7. For each of the four invariant-generating methods described below, we give an overview of the method and present some example invariants that the method produced.

The second major aspect to our approach is the automated verification of the results produced. This is important, because the theorems produced are too large to be checked by hand (given the number of the equivalence classes being considered). There are numerous lemmas which have to be proved in order to check an overall classification theorem, including: (a) checking that a property is invariant (b) checking that a particular algebraic structure satisfies the definition of a property (c) checking that a theorem covers all the equivalence classes for a particular size, etc. While some of these theorems pose little difficulty for automated theorem provers, we have found that other theorems are beyond the capabilities of state of the art provers. For this reason, we have experimented with numerous theorem proving systems and in some cases we have resorted to specifying the theorem as a satisfiability problem and using a SAT-solver. We have also used computer algebra techniques to simplify the problems being solved. In Section 2, we describe how we combine these various reasoning systems with the invariant generation methods to derive and verify classification results for a given size of a given algebraic domain over a given equivalence relation. We conclude by presenting three of the classification theorems in full, and describing some future directions for our work.

2. System overview

In [5], we have presented a bootstrapping procedure that constructs fully verified classification theorems for algebraic structures of fixed, finite order with respect to a given equivalence relation. In this section, we briefly outline the technique. The algorithm starts with only the basic axioms of a particular algebraic structure, successively computes properties to separate non-equivalent structures, and returns a set of unique distinguishing properties for all equivalent classes together with representant structures.

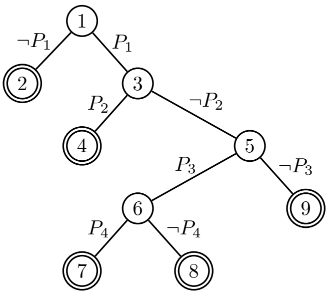
Schematically, the bootstrapping procedure works as in Figure 1. We see that the procedure takes a set of properties, $\mathcal{P}$, a cardinality, n, and an equivalence relation, $\sim$, as input. It returns a set, $\mathcal{I}$, consisting of algebraic structures together with sets of properties that uniquely define equivalence classes with respect to $\sim$. Thus, the set $\mathcal{I}$ represents the desired classification theorem. $\mathcal{I}$ is constructed iteratively, by first generating an algebraic structure Q of order n satisfying the initial properties $\mathcal{P}$ (step 1 in Figure 1). If we can show that Q together with $\mathcal{P}$ already forms an equivalent class then we are done (steps 5& 6). If the proof of this fails, we generate a structure Q' which satisfies the same properties as Q but that is not equivalent to Q. For the two structures we then compute a discriminant, i.e. a property invariant under $\sim$ such that it holds for Q but not for Q'. We then repeat the process for Q with properties $\mathcal{P} \cup \{P\}$ and Q' with $\mathcal{P} \cup \{\neg P\}$. The bootstrapping procedure successively generates structures and refines discriminating properties until the full set of equivalence classes is computed.

Bootstrapping (n, $\mathcal{P}$, $\sim$)

1: Generate structure Q of cardinality n satisfying $\mathcal{P}$.
2: Let $\mathcal{S} = \{(Q, \mathcal{P})\}$, $\mathcal{I} = \{\}$.
3: **Repeat**
4: Pick $(Q, \mathcal{P}) \in \mathcal{S}$
5: **If** $(Q, \mathcal{P})$ represents an equivalence class for order n under $\sim$ **then**
6: $\mathcal{I} = \{(Q, \mathcal{P})\} \cup \mathcal{I}$
7: **Else**
8: Generate Q' satisfying $\mathcal{P}$, such that $Q \not\sim Q'$
9: Find property P discriminating Q and Q' under $\sim$
10: $\mathcal{S} = \mathcal{S} \cup \{(Q, \mathcal{P} \cup \{P\}), (Q', \mathcal{P} \cup \{\neg P\})\}$
11: **Until** $\mathcal{S} \neq \emptyset$
12: Return $\mathcal{I}$.

FIGURE 1: Schematic overview of the bootstrapping algorithm

Technically, the bootstrapping procedure generates a binary decision tree, where leaf nodes correspond to the equivalence classes and each inner node represents branching with respect to a discriminating property. As an example, we discuss the isomorphism classification theorem for quasigroups of order 3. The decision tree as well as the five isomorphism class representants are given in Figure 2. The leaves 2, 4, 7, 8, and 9 of the tree are the isomorphism classes with respective representants Q_2, Q_4, Q_7, Q_8 and Q_9, respectively. The properties labelling the edges of the tree correspond to the discriminating properties. Thus the properties uniquely determining the isomorphism class represented by a leaf node correspond to the conjunction of the properties given on the path from the leaf to the root. The overall classification theorem then corresponds to a disjunction of conjunctions of discriminating properties.



$P_1 \equiv \exists b. b \circ b = b$
$P_2 \equiv \forall b. b \circ b = b$
$P_3 \equiv \exists b. \forall c. b \circ c \neq c$
$P_4 \equiv \exists b. \forall c. c \circ b \neq c$

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

Q_4	a	b	c
a	a	c	b
b	c	b	a
c	b	a	c

Q_7	a	b	c
a	a	b	c
b	b	c	a
c	c	a	b

Q_8	a	b	c
a	a	b	c
b	c	a	b
c	b	c	a

Q_9	a	b	c
a	a	c	b
b	b	a	c
c	c	b	a

FIGURE 2: Decision tree for the classification problem of order 3 quasigroups

The bootstrapping procedure is a framework that combines a host of reasoning techniques which play their part in achieving the overall goal. While the general framework is implemented in a Lisp environment, it relies on third party systems to generate algebraic structures and discriminants, and to verify the construction of the decision tree at each step. We give a brief overview on the methodologies used in the different steps of the algorithm. For a more detailed description of the techniques and problem formalisations, we refer the reader to [5], [13], [20].

Generating algebraic structures. We use model generation to construct algebraic structures. Model generation systems take as input a set of axioms, within which various variables are universally or existentially quantified in first order logic. The aim of model generation systems is to find a complete set of instantiations of the variables so that the axioms are satisfied. In step 1, the algorithm calls a model generator with the input axioms to obtain a structure that satisfies them. In step 8, a model generator is not only supplied with the properties the desired structure should satisfy, but also with an explicit formalisation of the structure it should not be equivalent to. We currently use the Sem [22], Finder [18] and Mace [11] systems for model generation.

Generating discriminants. The approach to constructing discriminating properties, necessary in step 9 of the bootstrapping procedure, varies from equivalence relation to equivalence relation. When dealing with the isomorphism relation, we treated the generation of a discriminant for a pair of algebras as a machine learning problem, and successfully applied automated theory formation [3] and inductive logic programming [6] with the HR system to solve such problems by finding first order invariants, as described in Sections 3 and 4. While these first order properties worked well as isomorphism invariants, we found that they did not discriminate between non-isotopic pairs of algebraic structures. In light of this, we developed bespoke methods for generating isotopy invariants, as described in Sections 5 and 6.

Verifying properties. Throughout the bootstrapping procedure, all the results from third party systems are independently verified by first order automated theorem provers. Thus, for a given discriminant P and two algebras Q and Q', we show that (1) P is a proper discriminant for the equivalence relation E [which means that if Q and Q' differ with respect to the property, then they cannot be members of the same equivalence class], (2) P holds for Q, and (3) P does not hold for Q'. Proving these properties explicitly guarantees the overall correctness of the constructed decision tree. The proofs themselves are generally not very challenging, and we have experimented with several provers. We generally employ the Spass [21], Vampire [16], and E [17] automated theorem provers for these tasks.

Verifying equivalence classes. The most difficult verification problems which occur during the classification process involve showing that a given node of the classification tree forms an equivalence class with respect to the equivalence relation under consideration. In other words, we need to verify that the tree cannot be expanded any further and that we do indeed have a leaf node. More formally, we need to prove that, for a particular set of properties P, all algebraic structures of cardinality n which satisfy P, are equivalent, and every member of the equivalence class satisfies P. These types of proofs are necessary to fully verify the completeness of the classification tree. Although the theorems are essentially second order, because we work in a finite domain, they can be expressed as propositional logic problems by enumerating all possible equivalence mappings for structures of cardinality n and thus can be made accessible to automated theorem proving systems. We have been particularly successful using satisfiability solvers, which are akin to model generators, but have more restricted representation requirements for the axioms to satisfy. In particular, each axiom must be a disjunction of Boolean variables, and so the axiom set is expressed in conjunctive normal form. Given this restriction, more powerful solving techniques are available to SAT-solvers. In addition, we have employed some so-called SMT solvers, which extend satisfiability solving by enabling the usage of additional decision procedures for particular theories. We have used the zChaff solver [15], DPLLT [8], and CVC-3 [2] satisfiability solvers.

While using SAT solvers increases the power of our algorithm, if translated naively, many of the proof problems would still be beyond the capabilities of state of the art systems. To enable us to solve these problems, we implemented some computer algebra algorithms in GAP [9] to reduce their complexity. For example, when showing that a particular set of properties constitutes an isomorphism class for structures of order n, the formulation of the theorem in propositional logic essentially amounts to enumerating and checking all possible bijective mappings between two structures of size n. Thus the number of mappings to consider grows quickly and to reduce it we use GAP to compute a generating system for the representant of the isomorphism class in question, thereby enabling us to reduce the number of bijective mappings to consider those on the generators alone. For the related isotopy problem, i.e., proving that all loops with a particular property are isotopic to each other, we have developed a similar technique. However, since generating systems are not invariant under isotopy, we instead generate all fg-isotopes of the given equivalence class representant and then show that every loop in the isotopy class has to be isomorphic to one of the fg-isotopes, which enables us to again reduce the number of mappings to consider to only those on the respective generating systems. For more details of these techniques see [20].

3. First-order isomorphism invariants

As we saw in Section 2, our automated approach to generating classification theorems relies heavily on a method to generate a discriminating property when given two example structures. That is, for instance, given two examples of non-equivalent loops, we need a method to determine a property which not only discriminates between the two loops but is sufficiently general enough that we can *prove* that two structures which differ according to this discriminant cannot be equivalent. Stated in this fashion, this is an instance of a machine learning problem, and hence our first approach to generating invariants used our machine learning system, HR, which is described in detail in [3].

In the application described here, HR starts with the two example algebraic structures and some background concepts describing them: the multiplication table (loops, quasigroups and groups), plus the concept of the identity element (loops and groups) and the concept of inverse elements (groups). HR then invents new concepts from old ones using a number of production rules. For instance, HR might use the *match* production rule to invent the concept of idempotent elements in loops (x s.t. $x = x \circ x$). It then might use the *exists* production rule to invent the concept of loops where there is such an idempotent element, followed by the *negate* production rule to invent the concept of loops with no idempotent elements. In this way, HR is able to produce a theory containing such concepts, and if the production rules are restricted, then the concept definitions will be expressed in first order logic (in the syntax of the Otter theorem prover [12]). Given the definitions of identity and inverse elements, each invariant is therefore simply an expression of a sequence of multiplication terms which only one structure has.

While HR has the ability to tailor its search to find discriminating concepts (as described in [4]), for the experiments described here, we simply ran HR with an exhaustive search until it found a single discriminating property, at which stage it outputs this result and terminates. The reason for this style of search is that it means that HR will consider simpler concepts for discriminants before more complex ones, which is important, as the concepts will be used later in numerous proofs. One such proof is to show that the property HR finds is indeed an invariant, which can be expressed in first order logic and hence first order resolution theorem provers can be used to prove these results by refutation. As an example, consider the proof that the property of all elements in a quasigroup being idempotent is invariant under isomorphism. Paraphrasing from the refutation proof of this found by a prover, let $(G, \circ), (H, \star)$ be quasigroups with isomorphism $\varphi : G \to H$. If $\forall g \in G (g \circ g = g)$ then we also have for every $h \in H$: $h \star h = h$. This is shown by first assuming that there is a $h \in H$ such that $h \star h \neq h$. Since G and H are isomorphic, there exists a unique $g \in G$ with $\varphi(g) = h$. Then we have $h \star h = \varphi(g) \star \varphi(g) = \varphi(g \circ g) = \varphi(g) = h$, which contradicts the assumption that $h \star h \neq h$, hence proving the invariant nature of the property.

4. Isomorphism invariants from counting sets of elements

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., $C_2 \times C_4$. In order to extend our classification approach, we have looked at the automatic generation of parameterisations of finite algebraic structures. This is motivated by a desire to produce classification theorems which apply not only to single orders of algebraic structures, but to be homogeneous across orders.

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. $x = x^{-1}$). Counting set sizes is an important tool in producing classification results. Moreover, 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. We present here the results of some initial experimentation with this approach, which has yet to be fully implemented into the bootstrapping algorithm.

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 (as described below).

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 have been given in Section 3, but 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 that the *Element-types* column above describes the total number of element-types produced by HR, while the *Classifiers* columns describe the number of those which were used in the parameterisation. The 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 orders 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 $\exists\, y\ (y * x = \mathrm{id} \wedge y * y = x)$. We consider it 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. We present the full classification theorem achieved for quasigroups of size 4 in Section 7.

Domain and (Orders)	Classes	Achieved	Steps	Element Types	Classifiers	Time (s)
Groups (1-8)	14	14	1000	32	3	28
Loops (1-5)	11	11	1000	32	3	10
Loops (1-6)	120	86	40000	736	11	2903
Quasigroups (1-4)	42	42	10000	523	5	215

TABLE 1: Parameterisation details in loop, group and quasigroup theory

To prove that the conjunctions of set sizes represent classifying concepts, we first translate the set-size properties into full first order logic by expressing the counting argument in a formal way. For instance, we define the property of having two self-inverse elements (in group theory) as:

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

We then need to solve 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

$$\forall A_1, A_2.\, \mathcal{P}(A_1) \wedge \mathcal{P}(A_2) \wedge P(A_1) \wedge \neg P(A_2) \implies A_1 \not\cong A_2,$$

where $\mathcal{P}$ describes the axiomatic properties of the algebraic structures and P is the discriminant under examination. 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 [21]. Problems of type (2) are less trivial since they are essentially second order theorems: we have to show that all algebraic structures that have property P are also isomorphic to the representant (which we call A_R), i.e.:

$$\forall A_\bullet [\mathcal{P}(A) \wedge P(A)] \implies [\exists \phi_\bullet \text{bijective}(\phi) \wedge \text{homomorphic}(\phi) \wedge \phi(A) = A_R].$$

However, since we are in a finite domain, we can explicitly formulate the problem in propositional logic: We give A_R in terms of its elements and multiplication table and then formulate all possible bijective mappings from an arbitrary structure A onto the elements of A_R. However, since the number of mappings to consider is $n!$, where n is the size of the structures A and A_R, 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 A_R, i.e., a set of elements that can generate all other elements of the structure together with all generating equations. While the problem formulation can still be relatively lengthy, we found that we could solve problems up to size 8 using the CVC-3 system [2]. 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 well as the majority of the 86 loops 6 problems. For groups up to size 8, however, our system failed to produce three problem formulations due to their size. On a positive note, all problems for which a formulation could be produced were shown to be correct by CVC-3 in less than 1 minute. We are currently optimising our routines to more efficiently produce larger problem formalisations, and we expect to solve the size 8 problems in due course.

5. Equational isotopy invariants

Our first method to obtain isotopy invariants for loops works by adapting Falconer's concepts of derived and universal identities presented in [7] to our needs. So far, it was sufficient to regard quasigroups and loops respectively as sets with a single operation. However, in order to follow Falconer's construction we will now define the two additional operations $\backslash$ and $/$.

Let $(Q, \circ)$ be a quasigroup, then we define two operations $\backslash$ and $/$ on Q such that:

(1) $x \cdot (x \backslash y) = y$ and $x \backslash (x \cdot y) = y$;
(2) $(y/x) \cdot x = y$ and $(y \cdot x)/x = y$.

Given a word w in Q, define its isotopically related word $\overline{w}$ by recursively applying the following transformations:

(1) if $w = x$, then $\overline{w} = x$;
(2) if $w = u \cdot v$, then $\overline{w} = (\overline{u}/y) \cdot (z \backslash \overline{v})$

where $y, z \in Q$ do not occur in w. For a given identity $w_1 = w_2$, where w_1, w_2 are words in Q we call the $\overline{w}_1 = \overline{w}_2$ equality a *derived identity*. A derived identity that is invariant under isotopy is called a *universal identity*.

From a logical point of view, an identity $w_1 = w_2$ is an equality where all variables occurring in the two words w_1 and w_2 are universally quantified. Moreover, the derived identity $\overline{w_1} = \overline{w_2}$ is constructed by introducing two new, universally quantified variables y and z. As an example of a universal identity, we consider the two following loops:

L_4	0	1	2	3	4	5
0	0	1	2	3	4	5
1	1	2	0	5	3	4
2	2	0	1	4	5	3
3	3	5	4	1	0	2
4	4	3	5	0	2	1
5	5	4	3	2	1	0

L_8	0	1	2	3	4	5
0	0	1	2	3	4	5
1	1	2	0	4	5	3
2	2	0	1	5	3	4
3	3	5	4	1	0	2
4	4	3	5	0	2	1
5	5	4	3	2	1	0

The following universal identity holds for L_4 but does not hold for L_8:

$$\begin{aligned}\forall x \centerdot \forall y_1 \centerdot \forall y_2 \centerdot (x/y_1) \cdot (((x/y_1) \cdot (y_2\backslash x)) \cdot (y_2\backslash x)) \\ = ((x/y_1) \cdot (y_2\backslash x)) \cdot ((x/y_1) \cdot (y_2\backslash x)).\end{aligned}$$

This universal identity was derived from the following loop identity:

$$\forall x \centerdot x \cdot ((x \cdot x) \cdot x) = (x \cdot x) \cdot (x \cdot x).$$

Falconer's concept of universal identity depends on deriving universal identities from loop identities that hold for the free loop. However, in our adaptation, we work in a strictly finite setting in order to derive universal identities that can be used as isotopy invariants to discriminate between loops. We have therefore devised the following algorithmic approach to constructing universal identities:

(1) In a first step, our algorithm systematically generates simple identities, i.e. universally quantified equations of the form $w_1 = w_2$.
(2) It then checks whether a non-trivial loop of size 4 to 8 exists satisfying the identity. This is achieved with a finite model generator.
(3) If such a loop exists, the algorithm rewrites $w_1 = w_2$ to its corresponding derived identity $\overline{w_1} = \overline{w_2}$.
(4) The derived identity is then passed to a first order theorem prover to show that it is invariant under isotopy. If the theorem prover succeeds, then $\overline{w_1} = \overline{w_2}$ is indeed a universal identity.

The presented algorithm is very effective and to date we have generated 8,530 universal identities. Moreover, despite starting in a very small finite setting, the resulting universal identities are shown to be isotopy invariants independent of the order of the loops or whether they are finite. Nevertheless, using universal identities only to find discriminating properties is not necessarily sufficient as there is no theoretical result guaranteeing that for two given non-isotopic quasigroups there is always a discriminating universal identity. Moreover, finding a suitable identity amounts to a considerable search task. Thus we have developed a more goal-directed approach to constructing isotopy invariants using exhaustive counting arguments, as described in the next section.

6. Sub-block isotopy invariants

As a more reliable method to obtain isotopy invariants, we developed the necessary theoretical tools to generate invariants based on exhaustive counting arguments which examine properties of sub-blocks of loops.

Let $(G, \cdot)$ be a quasigroup, and let A and B be non-empty subsets of G. We adopt the usual notation for the set $A \cdot B$, namely, $A \cdot B = \{a \cdot b : a \in A \wedge b \in B\}$.

Lemma 1. *Let $(G, \cdot)$ be a quasigroup and let $(H, *)$ be a quasigroup that is isotopic to $(G, \cdot)$ under the bijections (α, β, γ). Then, for any non-empty subsets A and B of G, we have $|A \cdot B| = |\alpha(A) * \beta(B)|$.*

PROOF: Observe that since γ is a bijection, then $|\gamma(A \cdot B)| = |A \cdot B|$. It suffices then to show that $\gamma(A \cdot B) = \alpha(A) * \beta(B)$. But this follows immediately from the fact that for all $a \in A$ and $b \in B$, we have $\gamma(a \cdot b) = \alpha(a) * \beta(b)$. $\square$

When G is finite, one can interpret the elements of A (resp., B) as designating a subset of rows (resp., columns) in the multiplication table of G. The set $A \cdot B$ then consists of the elements where these rows and columns meet. The above result thus suggests the following notation:

Notation 2. Let $(G, \cdot)$ be a quasigroup of order n, and let i, j, k each be integers such that $1 \le i, j, k \le n$. Let $G(i, j, k)$ denote the set:

$$G(i, j, k) = \{(A, B) : A, B \subseteq G, |A| = i, |B| = j, |A \cdot B| = k\}.$$

Theorem 3. *Let $(G, \cdot)$ and $(H, *)$ be isotopic quasigroups of order n, and let i, j, k each be integers such that $1 \le i, j, k \le n$. Then $|G(i, j, k)| = |H(i, j, k)|$.*

PROOF: Note that the one-to-one correspondence between the collection of ordered pairs (A, B) such that $A, B \subseteq G$, $|A| = i$, $|B| = j$, and the corresponding collection of ordered pairs of subsets of H, is preserved under isotopy. The result now follows from Lemma 1. $\square$

Example 4. As an example for sub-block invariants, consider the following two loops:

L_{38}	0	1	2	3	4	5
0	0	1	2	3	4	5
1	1	2	5	0	3	4
2	2	5	0	4	1	3
3	3	4	1	5	2	0
4	4	0	3	1	5	2
5	5	3	4	2	0	1

L_{20}	0	1	2	3	4	5
0	0	1	2	3	4	5
1	1	2	0	5	3	4
2	2	0	1	4	5	3
3	3	4	5	1	2	0
4	4	5	3	0	1	2
5	5	3	4	2	0	1

Loop L_{38} contains 4 different 2×2 sub-blocks that contain exactly 2 distinct elements, i.e., we have $|L_{38}(2,2,2)| = 4$. In detail, the single sub-blocks are the following:

	0	2
0	0	2
2	2	0

	2	4
1	5	3
4	3	5

	1	3
2	5	4
3	4	5

	3	5	
4	1	2	
5	2	1	7

For loop L_{20} on the other hand, we have $|L_{20}(2,2,2)| = 0$ since it does not contain a single 2×2 sub-block with two distinct elements.

The above results form the basis for two more isotopy-invariants, which we now present: frequency tuples and patterns.

6.1 Frequency Tuples. Continuing with the notation above, fix an element $(A, B) \in G(i, j, k)$, and for each $g_h \in A \cdot B$, with $1 \leq h \leq k$, let

$$f(g_h) = |\{(a, b) \in A \times B : a \cdot b = g_h\}|.$$

In other words, $f(g_h)$ is the number of times that g_h appears in the block formed by A and B, henceforth referred to as the $A \cdot B$ block. We let $F(A, B) = (f(g_1), \ldots, f(g_k))$, and call this the (un-ordered) *frequency-tuple* of (A, B). If two such frequency-tuples F and F' are the same (up to order), then we write $F \approx F'$.

Lemma 5. *Let $(G, \cdot)$ and $(H, *)$ be isotopic quasigroups (under the bijections (α, β, γ)) of order n, and let i, j, k each be integers such that $1 \leq i, j, k \leq n$. If $(A, B) \in G(i, j, k)$, then $F(A, B) \approx F(\alpha(A), \beta(B))$.*

PROOF: In light of Theorem 3, it suffices to prove that, for every $g \in A \cdot B$, $f(g) = f(\gamma(g))$. But this equality follows immediately from the fact that if $a \cdot b = g$, then $\alpha(a) * \beta(b) = \gamma(g)$. □

Given this latest result, we adopt the following notation:

Notation 6. Let $(G, \cdot)$ be a quasigroup of order n, let i, j, k be integers such that $1 \leq i, j, k \leq n$, and let F be a frequency-tuple for some $(C, D) \in G(i, j, k)$. Then, let $G(i, j, k, F)$ denote the set:

$$G(i, j, k, F) = \{(A, B) \in G(i, j, k) : F(A, B) \approx F\}.$$

Theorem 7. *Let $(G, \cdot)$ be a quasigroup of order n, let i, j, k be integers such that $1 \leq i, j, k \leq n$, and let F be a frequency-tuple for some $(C, D) \in G(i, j, k)$. Furthermore, let $(H, *)$ be a quasigroup isotopic to $(G, \cdot)$. Then $|G(i, j, k, F)| = |H(i, j, k, F)|$.*

Proof: This is an immediate consequence of Lemma 5 and Theorem 3. □

Example 8. To illustrate the idea of frequencies, consider again the loop L_{38} together with the two 3×3 sub-blocks S_1 and S_2 given on the right:

L_{38}	0	1	2	3	4	5
0	0	1	2	3	4	5
1	1	2	5	0	3	4
2	2	5	0	4	1	3
3	3	4	1	5	2	0
4	4	0	3	1	5	2
5	5	3	4	2	0	1

S_1	3	4	5
0	3	4	5
1	0	3	4
2	4	1	3

S_2	0	4	5
1	1	3	4
2	2	1	3
3	3	2	0

Both sub-blocks contain the same number of distinct elements, namely 5. However, S_1 contains three elements $(0, 1, 5)$ once and two elements $(3, 4)$ three times, whereas S_2 contains two elements $(0, 4)$ once, two elements $(1, 2)$ twice, and one element (3) three times. Thus S_1 has the frequency tuple $(1, 1, 1, 3, 3)$ and S_2 has the frequency tuple $(1, 1, 2, 2, 3)$. Overall for L_{38} we have invariants $|L_{38}(2, 2, 2, (1, 1, 1, 3, 3))| = 4$ and $|L_{38}(2, 2, 2, (1, 1, 2, 2, 3))| = 52$.

6.2 Patterns. Given non-empty subsets A and B of a quasigroup $(G, \cdot)$, we look for patterns amongst the numbers of distinct elements within the respective sub-blocks. By this, we mean the following: Let $|A| = i$, $|B| = j$, and choose i', j' such that $1 \leq i' \leq i$ and $1 \leq j' \leq j$. Now for each k, $1 \leq k \leq n$, we let $AB(i', j', k) = \{(A', B') : A' \subseteq A,\ B' \subseteq B,\ |A'| = i',\ |B'| = j',\ |A' \cdot B'| = k\}$. Furthermore, let $p_k = |AB(i', j', k)|$. In other words, p_k is the number of $i' \times j'$ sub-blocks of the $A \cdot B$ block, that have precisely k distinct entries. We now let $\mathfrak{P}_{i',j'}(A, B) = (p_1, \ldots, p_n)$, and we call $\mathfrak{P}_{i',j'}(A, B)$ the $i' \times j'$ *pattern-tuple* of (A, B).

Lemma 9. *Let $(G, \cdot)$, $(H, *)$, (α, β, γ), i, j, k, n be as in Lemma 5, and let i', j' be integers such that $1 \leq i' \leq i$ and $1 \leq j' \leq j$. If $A, B \subseteq G$ such that $|A| = i$ and $|B| = j$, then $\mathfrak{P}_{i',j'}(A, B) = \mathfrak{P}_{i',j'}(\alpha(A), \beta(B))$.*

Proof: Note that for each k, $1 \leq k \leq n$, and for each $(A', B') \in AB(i', j', k)$, we have $|A' \cdot B'| = |\alpha(A') * \beta(B')|$, by Lemma 1. Now since α and β are bijections, $(A', B') \in AB(i', j', k)$ if and only if $(\alpha(A'), \beta(B')) \in \alpha(A)\beta(B)(i', j', k)$. The result follows. □

Following similar lines as previously, we introduce the following notation:

Notation 10. Let $(G,\cdot)$ be a quasigroup of order n, and let $\mathfrak{P}_{i',j'}$ be an $i'\times j'$ pattern-tuple of (C,D) for some $C,D\subseteq G$ such that $|C|=i$ and $|D|=j$ $(1\leq i,j\leq n)$, where integers i',j' are such that $1\leq i'\leq i$ and $1\leq j'\leq j$. We let $G(i,j,\mathfrak{P}_{i',j'})$ denote the set:

$$G(i,j,\mathfrak{P}_{i',j'})=\{(A,B):A,B\subseteq G,|A|=i,|B|=j,\mathfrak{P}_{i',j'}(A,B)=\mathfrak{P}_{i',j'}\}.$$

Theorem 11. *Let $(G,\cdot)$ be a quasigroup of order n, and let $\mathfrak{P}_{i',j'}$ be an $i'\times j'$ pattern-tuple of (C,D) for some $C,D\subseteq G$ such that $|C|=i$ and $|D|=j$ $(1\leq i,j\leq n)$, where integers i',j' are such that $1\leq i'\leq i$ and $1\leq j'\leq j$. Furthermore, let $(H,*)$ be a quasigroup isotopic to $(G,\cdot)$. Then $\left|G(i,j,\mathfrak{P}_{i',j'})\right|=\left|H(i,j,\mathfrak{P}_{i',j'})\right|$.*

PROOF: This follows immediately from Lemma 9. □

Example 12. We illustrate patterns with the example of loop L_{25} below in which we are interested in 2×2 pattern tuples within 4×4 sub-blocks. The particular sub-block S below contains exactly one 2×2 sub-block with exactly two distinct elements.

L_{25}	0	1	2	3	4	5
0	0	1	2	3	4	5
1	1	2	0	4	5	3
2	2	0	1	5	3	4
3	3	5	4	1	0	2
4	4	3	5	0	2	1
5	5	4	3	2	1	0

S	1	2	3	4
2	0	1	5	3
3	5	4	1	0
4	3	5	0	2
5	4	3	2	1

S'	2	4
2	1	3
5	3	1

The overall pattern tuple for S' is $\mathfrak{P}_{2,2}=(0,1,12,23,0,0)$. The invariant for L_{25} counting the number of 4×4 sub-blocks with a pattern-tuple $\mathfrak{P}_{2,2}=(0,1,12,23,0,0)$ is $|L_{25}(4,4,\mathfrak{P}_{2,2})|=18$.

Observe that, rather than considering the entire $i'\times j'$ pattern-tuple of (A,B), we could instead, for instance, focus on only one component at a time, which simplifies the resulting invariant properties. With this in mind, we let $\mathfrak{P}_{i',j'}(A,B)_{(k)}$ denote the k-th component of the ordered n-tuple $\mathfrak{P}_{i',j'}(A,B)$. It is obvious then that, in the context of Lemma 9, we have $\mathfrak{P}_{i',j'}(A,B)_{(k)}=\mathfrak{P}_{i',j'}(\alpha(A),\beta(B))_{(k)}$. This leads to some further notation:

Notation 13. Let $(G,\cdot)$ be a quasigroup of order n, and let i,j,i',j',k,p_k be integers such that $1\leq i,j,k\leq n$, $1\leq i'\leq i$, $1\leq j'\leq j$ and $p_k\geq 0$. We let (G,i,j,i',j',k,p_k) denote the set:

$$\begin{aligned}&(G,i,j,i',j',k,p_k)\\&\qquad=\{(A,B):A,B\subseteq G,|A|=i,|B|=j\ \text{ and }\ \mathfrak{P}_{i',j'}(A,B)_{(k)}=p_k\}.\end{aligned}$$

Note that we have effectively proved the following corollary:

Corollary 14. *Let $(G,\cdot),(H,*),i,j,i',j',k,n$ be as in Lemma 9, and let p_k be a non-negative integer. Then*

$$\left|(G,i,j,i',j',k,p_k)\right| = \left|(H,i,j,i',j',k,p_k)\right|.$$

We now combine the notions of patterns and frequencies, by first expanding the notation $AB(i',j',k)$, described at the beginning of this section. The idea is that, for a given $(A,B) \in G(i,j,\mathfrak{P}_{i',j'})$, we want to see how the frequency-tuples $F(A',B')$ are distributed, for each $(A',B') \in AB(i',j',k)$ $(1 \le k \le n)$. To this end, for a frequency-tuple F of size k, we let $AB(i',j',k,F) = \{(A',B') \in AB(i',j',k) : F(A',B') \approx F\}$. This effectively partitions the collection of $i' \times j'$ sub-blocks of the $A \cdot B$ block, according to both their number of distinct elements and their frequency-tuples. We refer to this partition as the $i' \times j'$ *frequency distribution* $\mathfrak{F}_{i',j'}(A,B)$ of the $A \cdot B$ block.

Now since each of these properties — number of distinct elements, frequency-tuples, and pattern-tuples — is preserved under isotopism, then it follows that the number of $i \times j$ blocks with both a given pattern-tuple $\mathfrak{P}_{i',j'}$ and a given frequency distribution $\mathfrak{F}_{i',j'}$, is likewise preserved under isotopism. We could, of course, extend these properties recursively, by looking at sub-blocks of sub-blocks, and so on. It is an interesting question whether such a recursive extension would suffice to completely classify all (finite) loops.

All of the invariants in this section can be computed straightforwardly by appropriate recursive algorithms, which we have implemented in the Lisp environment of the overall bootstrapping algorithm. The general idea is to recursively inspect two loops by exhaustively computing sub-block, frequency, and pattern invariants, and combinations thereof, until a discriminating property has been found.

7. Classification theorems

In total, we have produced 7 full and 2 partial isomorphism classifications, and two full isotopism classifications, as follows:

Isomorphism results:

- Quasigroups of order 3, 4 and 5
- Idempotent quasigroups of order 6
- Quasigroups of order 6 with the extra property $\exists x \centerdot \forall y \centerdot (y \circ x) \circ (x \circ y) = x$, which is a generalised form of the QG3 property: $\forall x \centerdot \forall y \centerdot (y \circ x) \circ (x \circ y) = x$
- Quasigroups of order 7 with the QG9 property: $\forall x \centerdot \forall y \centerdot (((y \circ x) \circ x) \circ x) = y$ (Partial)
- Loops of order 5 and 6
- Idempotent loops of order 7 (Partial)

Isotopism results:

- Loops of order 6 and 7.

Below, we present the full classification theorem for quasigroups of order 4 obtained using first order invariants discussed in Section 3; the full isotopy theorem for loops of order 6, using both the equational and sub-block invariants discussed in Sections 5 and 6 respectively; and the full classification theorem for quasigroups of order 1 to 4 using the counting invariants discussed in Section 4.

Isomorphism classification theorem for quasigroups of order 4

We are given the following properties of quasigroups:

P_1:$\forall b. \forall c. (b \circ (b \circ c)) = c$
P_2:$\forall b. \exists c. (c \circ c) = b$
P_3:$\forall b. ((b \circ b) \circ (b \circ b)) = (b \circ b)$
P_4:$\forall b. \forall c. (c \circ b) = (b \circ c)$
P_5:$\exists b. (b \circ b) = b$
P_6:$\forall b. \forall c. (c \circ b) = (b \circ c)$
P_7:$\forall b. \forall c. (c \circ b) = (b \circ c)$
P_8:$\forall b. ((b \circ b) \circ (b \circ b)) = (b \circ b)$
P_9:$\forall b. \exists c. ((c \circ b) \circ c) = b$
P_{10}:$\forall b. \exists c. (c \circ b) = c$
P_{11}:$\forall b. \exists c. (b \circ c) = c$
P_{12}:$\forall b. \exists c. ((c \circ b) \circ (c \circ b)) = c$
P_{13}:$\exists b. (b \circ b) = b$
P_{14}:$\forall b. \exists c. (c \circ b) = c$
P_{15}:$\exists b. (b \circ b) = b$
P_{16}:$\forall b. \exists c. (c \circ (b \circ c)) = b$
P_{17}:$\forall b. ((b \circ b) \circ b) = b$
P_{18}:$\forall b. (b \circ b) = b$
P_{19}:$\forall b. \exists c. (c \circ c) = b$
P_{20}:$\forall b. \exists c. \exists d. ((c \circ d) = b \wedge (\neg(d \circ d) = b))$
P_{21}:$\forall b. \forall c. ((\neg(b \circ c) = b) \vee (c \circ c) = c)$
P_{22}:$\forall b. \exists c. (c \circ b) = c$
P_{23}:$\exists b. (b \circ b) = b$
P_{24}:$\forall b. ((b \circ b) \circ (b \circ b)) = (b \circ b)$
P_{25}:$\exists b. (b \circ b) = b$
P_{26}:$\exists b. \exists c. ((b \circ c) = b \wedge ((\neg(b \circ b) = c) \wedge (c \circ b) = b))$
P_{27}:$\forall b. ((\neg(b \circ (b \circ b)) = b) \vee ((b \circ b) \circ b) = b)$
P_{28}:$\forall b. \forall c. ((\neg(b \circ c) = b) \vee ((b \circ b) = c \vee (\neg(c \circ b) = b)))$
P_{29}:$\forall b. \exists c. ((c \circ (c \circ b)) = b \wedge (\neg(b \circ c) = (c \circ b)))$
P_{30}:$\forall b. \exists c. ((\neg(c \circ (c \circ b)) = b) \wedge (b \circ c) = (c \circ b))$
P_{31}:$\forall b. \exists c. ((b \circ c) \circ (b \circ c)) = c$
P_{32}:$\forall b. \exists c. (c \circ b) = c$
P_{33}:$\forall b. ((b \circ b) \circ (b \circ b)) = (b \circ b)$
P_{34}:$\exists b. (b \circ b) = b$
P_{35}:$\exists b. \exists c. ((b \circ c) = c \wedge (\neg(b \circ b) = b))$
P_{36}:$\exists b. ((b \circ (b \circ b)) = b \wedge (\neg((b \circ b) \circ b) = b))$

Then quasigroups of order 4 are characterised up to isomorphism by one of the following conjunction of properties (and their negations):

$(P_1 \wedge P_3 \wedge P_6)$
$(P_1 \wedge P_3 \wedge \neg P_6)$
$(P_1 \wedge \neg P_3 \wedge P_7)$
$(P_1 \wedge \neg P_3 \wedge \neg P_7 \wedge P_{12})$
$(P_1 \wedge \neg P_3 \wedge \neg P_7 \wedge \neg P_{12} \wedge \neg P_{19} \wedge P_{25})$
$(P_1 \wedge \neg P_3 \wedge \neg P_7 \wedge \neg P_{12} \wedge \neg P_{19} \wedge \neg P_{25})$
$(P_1 \wedge \neg P_3 \wedge \neg P_7 \wedge \neg P_{12} \wedge P_{19})$
$(\neg P_1 \wedge P_2 \wedge \neg P_5 \wedge P_{10})$
$(\neg P_1 \wedge P_2 \wedge \neg P_5 \wedge \neg P_{10})$
$(\neg P_1 \wedge P_2 \wedge P_5 \wedge \neg P_{11})$
$(\neg P_1 \wedge P_2 \wedge P_5 \wedge P_{11} \wedge \neg P_{18})$
$(\neg P_1 \wedge P_2 \wedge P_5 \wedge P_{11} \wedge P_{18})$
$(\neg P_1 \wedge \neg P_2 \wedge \neg P_4 \wedge P_9 \wedge P_{16} \wedge \neg P_{17})$
$(\neg P_1 \wedge \neg P_2 \wedge P_4 \wedge \neg P_8 \wedge P_{13})$
$(\neg P_1 \wedge \neg P_2 \wedge P_4 \wedge \neg P_8 \wedge \neg P_{13})$
$(\neg P_1 \wedge \neg P_2 \wedge P_4 \wedge P_8 \wedge \neg P_{14} \wedge \neg P_{20})$
$(\neg P_1 \wedge \neg P_2 \wedge P_4 \wedge P_8 \wedge \neg P_{14} \wedge P_{20})$
$(\neg P_1 \wedge \neg P_2 \wedge P_4 \wedge P_8 \wedge P_{14})$

$(\neg P_1 \wedge \neg P_2 \wedge \neg P_4 \wedge \neg P_9 \wedge P_{15} \wedge \neg P_{21} \wedge \neg P_{26} \wedge \neg P_{32} \wedge \neg P_{35})$
$(\neg P_1 \wedge \neg P_2 \wedge \neg P_4 \wedge \neg P_9 \wedge P_{15} \wedge \neg P_{21} \wedge \neg P_{26} \wedge \neg P_{32} \wedge P_{35})$
$(\neg P_1 \wedge \neg P_2 \wedge \neg P_4 \wedge \neg P_9 \wedge P_{15} \wedge \neg P_{21} \wedge \neg P_{26} \wedge P_{32})$
$(\neg P_1 \wedge \neg P_2 \wedge \neg P_4 \wedge \neg P_9 \wedge P_{15} \wedge \neg P_{21} \wedge P_{26} \wedge \neg P_{33})$
$(\neg P_1 \wedge \neg P_2 \wedge \neg P_4 \wedge \neg P_9 \wedge P_{15} \wedge \neg P_{21} \wedge P_{26} \wedge P_{33} \wedge \neg P_{36})$
$(\neg P_1 \wedge \neg P_2 \wedge \neg P_4 \wedge \neg P_9 \wedge P_{15} \wedge \neg P_{21} \wedge P_{26} \wedge P_{33} \wedge P_{36})$
$(\neg P_1 \wedge \neg P_2 \wedge \neg P_4 \wedge \neg P_9 \wedge P_{15} \wedge P_{21} \wedge \neg P_{27} \wedge P_{28})$
$(\neg P_1 \wedge \neg P_2 \wedge \neg P_4 \wedge \neg P_9 \wedge P_{15} \wedge P_{21} \wedge P_{27} \wedge \neg P_{28})$
$(\neg P_1 \wedge \neg P_2 \wedge \neg P_4 \wedge \neg P_9 \wedge \neg P_{15} \wedge P_{22})$
$(\neg P_1 \wedge \neg P_2 \wedge \neg P_4 \wedge \neg P_9 \wedge \neg P_{15} \wedge \neg P_{22} \wedge \neg P_{29} \wedge P_{30})$
$(\neg P_1 \wedge \neg P_2 \wedge \neg P_4 \wedge \neg P_9 \wedge \neg P_{15} \wedge \neg P_{22} \wedge P_{29} \wedge \neg P_{30})$
$(\neg P_1 \wedge \neg P_2 \wedge \neg P_4 \wedge P_9 \wedge \neg P_{16} \wedge \neg P_{17} \wedge \neg P_{23})$
$(\neg P_1 \wedge \neg P_2 \wedge \neg P_4 \wedge P_9 \wedge \neg P_{16} \wedge \neg P_{17} \wedge P_{23})$
$(\neg P_1 \wedge \neg P_2 \wedge \neg P_4 \wedge P_9 \wedge \neg P_{16} \wedge P_{17} \wedge P_{24})$
$(\neg P_1 \wedge \neg P_2 \wedge \neg P_4 \wedge P_9 \wedge \neg P_{16} \wedge P_{17} \wedge \neg P_{24} \wedge P_{31})$
$(\neg P_1 \wedge \neg P_2 \wedge \neg P_4 \wedge P_9 \wedge \neg P_{16} \wedge P_{17} \wedge \neg P_{24} \wedge \neg P_{31} \wedge P_{34})$
$(\neg P_1 \wedge \neg P_2 \wedge \neg P_4 \wedge P_9 \wedge \neg P_{16} \wedge P_{17} \wedge \neg P_{24} \wedge \neg P_{31} \wedge \neg P_{34})$

Isotopism classification theorem for loops of order 6

We are given the following properties of loops:

P_1: $\forall x, y_1, y_2 \centerdot (x/y_1)\cdot(((x/y_1)\cdot(y_2\backslash x))\cdot(y_2\backslash x)) = ((x/y_1)\cdot(y_2\backslash x))\cdot((x/y_1)\cdot(y_2\backslash x))$
P_2: $\forall x, y_1, y_2 \centerdot (x/y_1)\cdot(((x/y_1)\cdot(y_2\backslash y_1))\cdot(y_2\backslash y_1)) = ((x/y_1)\cdot((x/y_1)\cdot(y_2\backslash x)))\cdot(y_2\backslash x)$
P_3: $\forall x, y_1, y_2 \centerdot (x/y_1) = ((x/y_1)\cdot(y_2\backslash x))\cdot(y_2\backslash x)$
P_4: $|G(3,3,4)| = 0$ P_5: $|G(3,3,3)| = 0$ P_6: $|G(2,3,3)| = 0$
P_7: $|G(2,2,2)| = 7$ P_8: $|G(2,3,3)| = 8$ P_9: $|G(2,2,2)| = 0$
P_{10}: $|G(2,2,2)| = 9$
P_{11}: $|G(4,4,\mathfrak{P}_{2,2})| = 0 \wedge \mathfrak{P}_{2,2} = (0,1,12,23,0,0)$
P_{12}: $|G(3,3,3)| = 8$ P_{13}: $|G(2,2,2)| = 5$ P_{14}: $|G(2,3,3)| = 4$
P_{15}: $|G(3,2,3)| = 4$ P_{16}: $|G(2,2,2)| = 4$ P_{17}: $|G(2,2,2)| = 11$

Then loops of order 6 are characterised up to isotopism by one of the following conjunctions of properties:

$(\neg P_1 \wedge \neg P_2)$
$(\neg P_1 \wedge P_2 \wedge P_3)$
$(\neg P_1 \wedge P_2 \wedge \neg P_3)$
$(P_1 \wedge P_4 \wedge P_5 \wedge \neg P_6 \wedge P_7)$
$(P_1 \wedge P_4 \wedge P_5 \wedge \neg P_6 \wedge \neg P_7)$
$(P_1 \wedge P_4 \wedge P_5 \wedge P_6 \wedge P_7)$
$(P_1 \wedge P_4 \wedge P_5 \wedge P_6 \wedge \neg P_7 \wedge P_8)$
$(P_1 \wedge P_4 \wedge P_5 \wedge P_6 \wedge \neg P_7 \wedge \neg P_8)$
$(P_1 \wedge P_4 \wedge \neg P_5 \wedge P_9)$
$(P_1 \wedge P_4 \wedge \neg P_5 \wedge \neg P_9 \wedge \neg P_{10})$
$(P_1 \wedge P_4 \wedge \neg P_5 \wedge \neg P_9 \wedge P_{10} \wedge P_{11})$
$(P_1 \wedge P_4 \wedge \neg P_5 \wedge \neg P_9 \wedge P_{10} \wedge \neg P_{11})$
$(P_1 \wedge \neg P_4 \wedge P_6 \wedge P_7)$
$(P_1 \wedge \neg P_4 \wedge P_6 \wedge \neg P_7 \wedge P_{12})$
$(P_1 \wedge \neg P_4 \wedge P_6 \wedge \neg P_7 \wedge \neg P_{12})$
$(P_1 \wedge \neg P_4 \wedge \neg P_6 \wedge \neg P_{13} \wedge P_{14} \wedge P_{15})$
$(P_1 \wedge \neg P_4 \wedge \neg P_6 \wedge \neg P_{13} \wedge P_{14} \wedge \neg P_{15})$
$(P_1 \wedge \neg P_4 \wedge \neg P_6 \wedge \neg P_{13} \wedge \neg P_{14})$
$(P_1 \wedge \neg P_4 \wedge \neg P_6 \wedge P_{13} \wedge P_{16})$
$(P_1 \wedge \neg P_4 \wedge \neg P_6 \wedge P_{13} \wedge \neg P_{16} \wedge \neg P_{14})$
$(P_1 \wedge \neg P_4 \wedge \neg P_6 \wedge P_{13} \wedge \neg P_{16} \wedge P_{14} \wedge P_{17})$
$(P_1 \wedge \neg P_4 \wedge \neg P_6 \wedge P_{13} \wedge \neg P_{16} \wedge P_{14} \wedge \neg P_{17})$

Isomorphism classification theorem for quasigroups up to order 4

We are given the following five functions which return coefficients of element types for quasigroups, Q:

$f_1(Q) = |\{b : b \in Q\}|$
$f_2(Q) = 0 \;\text{ if }\; \nexists x_{\centerdot} x * x = x \;\text{ or }\; f_2(Q) = |\{b : \exists d_{\centerdot} (d * d = b)\}|$ otherwise
$f_3(Q) = |\{b : \exists c_{\centerdot} \exists d_{\centerdot} (c * b = b * c = d \wedge c * d \neq b \wedge b * d \neq c)\}|$
$f_4(Q) = |\{b : \exists c_{\centerdot} \exists d_{\centerdot} (c * d = d * c = b \wedge b * c \neq d)\}|$
$f_5(Q) = |\{b : (b * b) * b = b\}|$

Then, quasigroups up to and including size 4 are characterised up to isomorphism by the following parameterisations:

1.1) $f_1 = 1$
2.1) $f_1 = 2$
3.1) $f_1 = 3 \wedge f_2 = 0$
3.2) $f_1 = 3 \wedge f_2 = 3 \wedge f_3 = 2$
3.3) $f_1 = 3 \wedge f_2 = 3 \wedge f_3 = 0$
3.4) $f_1 = 3 \wedge f_2 = 1 \wedge f_3 = 2$
3.5) $f_1 = 3 \wedge f_2 = 1 \wedge f_3 = 0$
4.1) $f_1 = 4 \wedge f_2 = 1 \wedge f_3 = 4$
4.2) $f_1 = 4 \wedge f_2 = 4 \wedge f_3 = 2$
4.3) $f_1 = 4 \wedge f_2 = 4 \wedge f_3 = 3$
4.4) $f_1 = 4 \wedge f_2 = 2 \wedge f_3 = 4$
4.5) $f_1 = 4 \wedge f_2 = 0 \wedge f_3 = 0$
4.6) $f_1 = 4 \wedge f_2 = 3 \wedge f_3 = 0$
4.7) $f_1 = 4 \wedge f_2 = 2 \wedge f_3 = 3$
4.8) $f_1 = 4 \wedge f_2 = 3 \wedge f_3 = 1$
4.9) $f_1 = 4 \wedge f_2 = 1 \wedge f_3 = 0 \wedge f_4 = 0$
4.10) $f_1 = 4 \wedge f_2 = 1 \wedge f_3 = 2 \wedge f_4 = 4$
4.11) $f_1 = 4 \wedge f_2 = 2 \wedge f_3 = 0 \wedge f_4 = 2$
4.12) $f_1 = 4 \wedge f_2 = 2 \wedge f_3 = 2 \wedge f_4 = 0$
4.13) $f_1 = 4 \wedge f_2 = 0 \wedge f_3 = 4 \wedge f_4 = 2$
4.14) $f_1 = 4 \wedge f_2 = 4 \wedge f_3 = 0 \wedge f_4 = 3$
4.15) $f_1 = 4 \wedge f_2 = 4 \wedge f_3 = 0 \wedge f_4 = 0$
4.16) $f_1 = 4 \wedge f_2 = 0 \wedge f_3 = 4 \wedge f_4 = 4$
4.17) $f_1 = 4 \wedge f_2 = 2 \wedge f_3 = 2 \wedge f_4 = 2$
4.18) $f_1 = 4 \wedge f_2 = 2 \wedge f_3 = 2 \wedge f_4 = 4$
4.19) $f_1 = 4 \wedge f_2 = 2 \wedge f_3 = 0 \wedge f_4 = 0$
4.20) $f_1 = 4 \wedge f_2 = 1 \wedge f_3 = 2 \wedge f_4 = 0$
4.21) $f_1 = 4 \wedge f_2 = 1 \wedge f_3 = 0 \wedge f_4 = 2$
4.22) $f_1 = 4 \wedge f_2 = 0 \wedge f_3 = 3 \wedge f_4 = 4$
4.23) $f_1 = 4 \wedge f_2 = 0 \wedge f_3 = 3 \wedge f_4 = 0$
4.24) $f_1 = 4 \wedge f_2 = 0 \wedge f_3 = 3 \wedge f_4 = 3$
4.25) $f_1 = 4 \wedge f_2 = 0 \wedge f_3 = 3 \wedge f_4 = 2$
4.26) $f_1 = 4 \wedge f_2 = 3 \wedge f_3 = 3 \wedge f_4 = 0$
4.27) $f_1 = 4 \wedge f_2 = 3 \wedge f_3 = 3 \wedge f_4 = 2$
4.28) $f_1 = 4 \wedge f_2 = 3 \wedge f_3 = 2 \wedge f_4 = 3$
4.29) $f_1 = 4 \wedge f_2 = 3 \wedge f_3 = 3 \wedge f_4 = 3$
4.30) $f_1 = 4 \wedge f_2 = 0 \wedge f_3 = 2 \wedge f_4 = 2 \wedge f_5 = 0$
4.31) $f_1 = 4 \wedge f_2 = 0 \wedge f_3 = 2 \wedge f_4 = 2 \wedge f_5 = 2$
4.32) $f_1 = 4 \wedge f_2 = 3 \wedge f_3 = 2 \wedge f_4 = 2 \wedge f_5 = 1$
4.33) $f_1 = 4 \wedge f_2 = 3 \wedge f_3 = 2 \wedge f_4 = 2 \wedge f_5 = 2$
4.34) $f_1 = 4 \wedge f_2 = 1 \wedge f_3 = 3 \wedge f_4 = 2 \wedge f_5 = 2$
4.35) $f_1 = 4 \wedge f_2 = 1 \wedge f_3 = 3 \wedge f_4 = 2 \wedge f_5 = 1$

8. Conclusions

We have presented the methods behind and results from a series of applications of automated reasoning techniques to the classification of loops and quasigroups. In particular, we have used machine learning techniques to find isomorphic invariants both as first order properties and as a numerical parameterisations over set sizes. Moreover, we have introduced novel methods for producing isotopic invariants, based on equational methods and the analysis of sub-blocks within algebraic structures. In addition, we have described the usage of numerous reasoning techniques, including first order resolution theorem proving, model generation, satisfiability solving and computer algebra methods in the automated verification of the classification theorems produced. These techniques have been combined into a bootstrapping approach which has been successfully used to produce novel full classification theorems for loops and quasigroups up to isomorphism and isotopy. In addition to increasing somewhat our understanding of these algebraic domains, this application has pushed the boundaries of what is achievable in automated mathematics.

In future work, we intend to address some bottlenecks in the bootstrapping process, in order to progress to classifying higher orders. In particular, we intend to explore the usage of more powerful theorem proving methods, and to re-introduce the machine learning method — with improvements — to the problem of finding isotopic invariants. Our ultimate aim is to add to the general classification of loops and quasigroups. To do this, we intend to automatically find *families* of such algebraic structures, where a family is parameterised by a pair $\langle S, b(n)\rangle$, where S is a set of properties that all members of the family have and which characterises them with respect to the isomorphism/isotopism equivalence relation, and $b(n)$ is a Boolean test on integers which, if positive for a particular n prescribes that the family will have a member of order n. This will be a significant challenge to automate, as it amounts to a second-order problem, and hence we will need to employ higher-order theorem provers.

References

[1] Alur R., Peled D., Eds., *Computer Aided Verification*, 16^{th} International Conference, CAV 2004, vol. 3114 of *LNCS*, Springer, Boston, MA, 2004.

[2] Barrett C., Berezin S., *CVC Lite: A new implementation of the cooperating validity checker*, in Alur and Peled [1], pp. 515–518.

[3] Colton S., *Automated Theory Formation in Pure Mathematics*, Springer, 2002.

[4] Colton S., Bundy A., Walsh T., *Automatic identification of mathematical concepts*, in Machine Learning: Proceedings of the 17th International Conference, 2000, pp. 183–190.

[5] Colton S., Meier A., Sorge V., McCasland R., *Automatic Generation of classification theorems for finite algebras*, in David Basin and Michael Rusinowitch, Eds., Automated Reasoning — 2nd International Joint Conference, IJCAR 2004, vol. 3097 of *LNAI*, Springer, Cork, Ireland, 2004, pp. 400–414,.

[6] Colton S., Muggleton S., *Mathematical applications of inductive logic programming*, Machine Learning **64** (2006), 25–64.

[7] Falconer E., *Isotopy invariants in quasigroups*, Trans. Amer. Math. Society **151** (1970), 511–526.

[8] Ganzinger H., Hagen G., Nieuwenhuis R., Oliveras A., Tinelli C., *DPLL(T): Fast decision procedures*, in Alur and Peled [1], pp. 175–188.

[9] *The GAP Group*, GAP – Groups, Algorithms, and Programming, Version 4.3, 2002, `http://www.gap-system.org`.

[10] Kronecker L., *Auseinandersetzung einiger Eigenschaften der Klassenanzahl idealer komplexer Zahlen*, Monatsbericht der Berliner Akademie, pp. 881–889, 1870.

[11] McCune W., *Mace4 Reference Manual and Guide*, Argonne National Laboratory, 2003. ANL/MCS-TM-264.

[12] McCune W., *Otter 3.3 Reference Manual*, Technical Report ANL/MCS-TM-263, Argonne National Laboratory, 2003.

[13] Meier A., Sorge V., *Applying SAT solving in classification of finite algebras*, J. Automat. Reason. **35** (2005), no. 1–3, 201–235.

[14] Mitchell T., *Machine Learning*, McGraw Hill, New York, 1997.

[15] Moskewicz M., Madigan C., Zhao Y., Zhang L., Malik S., *Chaff: Engineering an efficient SAT solver*, in Proc. of the 39^{th} Design Automation Conference (DAC 2001), Las Vegas, 2001, pp. 530–535.

[16] Riazanov A., Voronkov A., *Vampire 1.1*, in Rejeev Goré, Alexander Leitsch, and Tobias Nipkow, Eds., Automated Reasoning — 1st International Joint Conference, IJCAR 2001, vol. 2083 of *LNAI*, Springer, Siena, Italy, 2001, pp. 376–380.

[17] Schulz S., *E: A Brainiac theorem prover*, Journal of AI Communication **15** (2002), no. 2–3, 111–126.

[18] Slaney J., *FINDER, Notes and Guide*, Center for Information Science Research, Australian National University, 1995.

[19] Slaney J., Fujita M., Stickel M., *Automated reasoning and exhaustive search: Quasigroup existence problems*, Comput. Math. Appl. **29** (1995), 115–132.

[20] Sorge V., Meier A., McCasland R., Colton S., *The automatic construction of isotopy invariants*, in Third International Joint Conference on Automated Reasoning, 2006, pp. 36–51.

[21] Weidenbach C., Brahm U., Hillenbrand T., Keen E., Theobald C., Topic D., *SPASS Version 2.0*, in A. Voronkov, Ed., Proc. of the 18th International Conference on Automated Deduction (CADE–18), vol. 2392 of *LNAI*, Springer, Berlin, 2002, pp. 275–279.

[22] Zhang J., Zhang H., *SEM User's Guide*, Department of Computer Science, University of Iowa, 2001.

School of Computer Science, University of Birmingham, UK

E-mail: V.Sorge@cs.bham.ac.uk
URL: `http://www.cs.bham.ac.uk/~vxs`

Department of Computing, Imperial College London, UK

E-mail: sgc@doc.ic.ac.uk
URL: `http://www.doc.ic.ac.uk/~sgc`

School of Informatics, University of Edinburgh, UK

E-mail: rmccasla@inf.ed.ac.uk
URL: `http://www.inf.ed.ac.uk/~rmccasla`

DFKI GmbH, Saarbrücken, Germany

E-mail: ameier72@web.de
URL: `http://www.ags.uni-sb.de/~ameier`

(*Received* October 16, 2007, *revised* November 24, 2007)