Automatic Construction and Verification of Isotopy Invariants

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

[1] School of Computer Science, University of Birmingham, UK
V.Sorge@cs.bham.ac.uk
http://www.cs.bham.ac.uk/~vxs
[2] DFKI GmbH, Saarbrücken, Germany
ameier72@web.de
http://www.ags.uni-sb.de/~ameier
[3] School of Informatics, University of Edinburgh, UK
rmccasla@inf.ed.ac.uk
http://www.inf.ed.ac.uk/~rmccasla
[4] Department of Computing, Imperial College London, UK
sgc@doc.ic.ac.uk
http://www.doc.ic.ac.uk/~sgc

Abstract. We extend our previous study of the automatic construction of isomorphic classification theorems for algebraic domains by considering the *isotopy* equivalence relation, which is of more importance than isomorphism in certain domains. This extension was not straightforward, and we had to solve two major technical problems, namely generating and verifying isotopy invariants. Concentrating on the domain of loop theory, we have developed three novel techniques for generating isotopic invariants, by using the notion of universal identities and by using constructions based on substructures. In addition, given the complexity of the theorems which verify that a conjunction of the invariants form an isotopy class, we have developed ways of simplifying the problem of proving these theorems. Our techniques employ an intricate interplay of computer algebra, model generation, theorem proving and satisfiability solving methods. To demonstrate the power of the approach, we generate an isotopic classification theorem for loops of size 6, which extends the previously known result that there are 22. This result was previously beyond the capabilities of automated reasoning techniques.

1 Introduction

The classification of abstract algebraic objects such as groups and rings has been a major driving force in pure mathematics. For the majority of algebraic domains, however, full classifications have been elusive (with notable exceptions being Abelian groups, finite simple groups and Abelian quasigroups [17]). This is partially due to the sheer number of classes in domains such as quasigroups, where the axioms are not particularly restrictive. Due to this volume of classes,

⋆ The author's work was supported by EPSRC MathFIT grant GR/S31099.

U. Furbach and N. Shankar (Eds.): IJCAR 2006, LNAI 4130, pp. 36–51, 2006.
© Springer-Verlag Berlin Heidelberg 2006

automatic techniques have much potential to add to mathematical knowledge. In particular, enumeration techniques have been used to count the instances of certain finite algebras [11] and model generating and constraint solving techniques could solve open existence problems [19]. Moreover, using first order theorem proving, McCune et. al have generated single axiom representations for numerous algebraic domains [9].

We are more interested in qualitative rather than quantitative results. To this end, we have developed a bootstrapping algorithm for the construction of classification theorems in finite algebraic domains. While it is largely generic, its power relies upon the automatic discovery of algebraic invariants which are able to discriminate between members of different classes defined by the equivalence relation under consideration. When considering isomorphism equivalences, we were able to employ machine learning techniques to generate the invariants, and this led to novel theorems which achieve classifications up to isomorphism of quasigroups and loops of small order [4]. We provide a brief overview of the bootstrapping algorithm in Sec. 2.

We present here the application of the bootstrapping algorithm to the production of classification theorems up to *isotopism*, an equivalence relation which is of greater importance than isomorphism in certain domains, in particular algebraic loop theory. Unfortunately, the machine learning approach did not suffice for this application, as finding isotopy invariants is a much more complex task. Hence, concentrating on the domain of loop theory, we have developed new methods for generating isotopy invariants, as described in Sec. 3. Firstly, using results from the literature, we show how universal identities can be used to generate invariants via an intricate interplay of model generation and theorem proving. Secondly, we present two new sets of invariants derived by systematically examining substructures of loops, and we describe their construction, which uses symbolic computation techniques.

As described in Sec. 4, it has also been a challenge to automatically verify that a conjunction of invariant properties defines an isotopy class (another important aspect of the bootstrapping algorithm). For this reason, we have developed methods for simplifying the problems with computer algebra, before providing proof via a satisfiability solver. As a by-product, these methods significantly simplify the task of generating non-isotopic models. Having solved the problems of generating and verifying isotopic invariants, we have employed the bootstrapping algorithm to generate new results. In particular, in Sec. 5, we present an isotopic classification theorem for loops of order 6, which extends the known enumeration theorem by providing a full set of classification properties.

2 Background

As described in [4], we have developed a bootstrapping algorithm for generating classification theorems in algebraic domains of pure mathematics. Moreover, we have applied this approach to constructing new classification results with respect to isomorphism mainly in the algebraic domains of quasigroups and loops. In this section, we briefly outline this method, and summarise our previous results.

The bootstrapping algorithm takes a set of properties, $\mathcal{P}$, a cardinality, n, and an equivalence relation, E, as input. It returns a decision tree that contains the classification theorem for the algebraic structures of order n that satisfy $\mathcal{P}$ with respect to E, as well as a set of representants for each equivalence class. During the construction, a set of theorems are proved, which – taken together – prove the correctness and the completeness of the classification theorem.

In detail, the method proceeds as follows. Firstly, it initialises a decision tree with root node $\mathcal{N}$ labelled with the properties $\mathcal{P}$. We denote the properties that a node is labelled by as $\mathcal{P}_{\mathcal{N}}$. The algorithm then works iteratively, starting with the root node and moving on to successive nodes in the tree. It constructs an example of an algebraic structure of order n satisfying $\mathcal{P}_{\mathcal{N}}$. When no example can be produced, the algorithm will prove that no structure of size n with properties $\mathcal{P}_{\mathcal{N}}$ can exist. When an example does exist, one of two things happen, either: (1) the process shows that the node represents an equivalence class with respect to E, i.e., it proves that all structures of order n that satisfy the properties $\mathcal{P}_{\mathcal{N}}$ are equivalent to each other under E, or (2) the process constructs another algebraic structure satisfying $\mathcal{P}_{\mathcal{N}}$, which is not equivalent to the first one. Note that cases (1) and (2) are mutually exclusive and can be performed in parallel. For case (2), the algorithm computes a discriminating property P for the two structures, such that P holds for one structure and $\neg P$ holds for the other. P is then used to further expand the decision tree by adding two new nodes $\mathcal{N}'$ and $\mathcal{N}''$ below $\mathcal{N}$, with labels $\mathcal{P}_{\mathcal{N}'} = \mathcal{P}_{\mathcal{N}} \cup \{P\}$ and $\mathcal{P}_{\mathcal{N}''} = \mathcal{P}_{\mathcal{N}} \cup \{\neg P\}$ respectively.

The algorithm then iterates over these nodes and adds to the tree accordingly. After new nodes have been created for each of these nodes, the above steps are carried out again. The algorithm terminates once no more expansions can be applied. Leaf nodes either represent equivalence classes or are empty, i.e., no structure exists with the properties given in the node. The final classification theorem comprises the disjunction of the properties which label the leaf nodes.

The bootstrapping algorithm is a framework that combines a host of reasoning techniques which play their part in achieving the overall goal. In particular, it relies on third party systems to generate algebras and discriminants, and to verify the construction of the decision tree at each step. We use the following methodologies in the different steps of the algorithm:

Generating Algebras. We use model generation to construct algebras. In the first step, the algorithm calls a model generator to return an algebra corresponding to the input axioms. Throughout the process, model generators are used to construct algebras which are not equivalent to a given algebra. For the experiments described in [4], we used the model generator Mace [10], but we have replaced this by Sem [22] and Finder [18], as they are more effective in our domain. While Sem is generally the more powerful of the two, it has weaknesses when dealing with function symbols of arity greater than 2, which can be introduced when we employ Skolemisation techniques (see [12] for details).

Generating Discriminants. The approach to constructing discriminating properties 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 [5] to such problems.

Verifying Properties. Throughout the bootstrapping procedure, all the results coming 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 Spass [21] for these tasks.

Verifying Equivalence Classes. The most difficult verification problems occurring during the classification process involve showing that a given node forms an equivalence class with respect to the equivalence relation under consideration. More formally, we need to prove that, for a particular set of properties of a node, P, all algebras of cardinality n, which satisfy P, are equivalent, and every member of the equivalence class satisfies P. These types of proof are necessary to fully verify the completeness of a decision 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 made accessible to ATP systems. In our original experiments, described in [4], we used Spass for these problems, as it was the only system which coped with the massive clause normalisations required (cf. [20]). For later experiments, we replaced Spass by state of the art SAT solvers and developed a range of encoding techniques for a diverse range of systems (cf. [12]). Currently we have integrated zChaff [13] that can handle pure boolean SAT problems, the DPLLT [7] system, which can handle ground equations, and CVClite [2], which can also deal with finite quantification. 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 [8] that exploit some mathematical domain knowledge to reduce complexity.

In our experiments with isomorphism as the equivalence relation, we mainly concentrated on the domain of quasigroups and loops. A *quasigroup* is a non-empty set G together with a binary operation $\circ$ that satisfies the property $\forall a, b \in G\centerdot\ (\exists x \in G\centerdot x \circ a = b) \wedge (\exists y \in G\centerdot a \circ y = b)$. This property is often called the Latin Square property and has the effect that every element of G appears exactly once in every row and every column of the multiplication table of $\circ$. A *loop* is a quasigroup that contains a unit element, i.e., an element e such that $\forall x \in G\centerdot x \circ e = e \circ x = x$. We generated novel isomorphism classification theorems for quasigroups of orders 3 to 5 and loops of order 4 to 6, a partial classification of loops of order 7, as well as some classification theorems for quasigroups of orders 6 and 7 with additional special properties, e.g., idempotency. The largest decision tree so far is the full isomorphism classification of quasigroups of size 5. This contains 2875 nodes and 1411 isomorphism classes, but is relatively shallow, with a maximum depth of 23. Its completion took more than four months of processing time. For more details see http://www.cs.bham.ac.uk/˜vxs/quasigroups/.

3 Generating Isotopy Invariants

The isotopy equivalence relation is of considerable importance in quasigroup theory, hence we concentrate on isotopy in this paper. It is defined as follows:

Definition 1. *We say two quasigroups* $(G,\cdot)$ *and* $(H,*)$ *are* isotopic *to each other — or* G *is an* isotope *of* H *— if there are bijective mappings* α,β,γ *from* G *to* H *such that for all* $x,y \in G$, $\alpha(x) * \beta(y) = \gamma(x \cdot y)$ *holds.*

Isotopy is an equivalence relation and the classes induced by it are called isotopism classes. It is a generalisation of isomorphism, since G and H are isomorphic if $\alpha = \beta = \gamma$. In other words, while two quasigroups can be isotopic but not necessarily isomorphic to each other, all members of an isomorphism class belong to the same isotopism class. Importantly, every quasigroup is isotopic to a loop, which we call its loop-isotope [14]. Thus, in certain respects, we can restrict the classification of quasigroups to just the classification of loops.

As mentioned above, an important function of the bootstrapping algorithm is to generate invariant properties which enable us to discriminate between two algebras belonging to different equivalence classes. For instance, when dealing with the isomorphism equivalence relation, if one example of an algebra was commutative (i.e., $\forall x, y_{\bullet}(x*y = y*x)$) and another was not, then they could not belong to the same isomorphism class. Our machine learning approach was able to generate such properties, given background concepts including the multiplication operation. Unfortunately, such simple properties do not enable us to distinguish between members of different isotopy classes. For example, commutativity is not an isotopy invariant, as the isotopism (α, ι, ι), where ι is the identity mapping and $\alpha \neq \iota$, can map a commutative quasigroup to a non-commutative isotope. Due to the more complicated nature of isotopy invariants, initial experiments with the learning system failed to identify any suitable isotopy invariants.

In light of this failure, we developed bespoke methods for generating three types of isotopy invariants that are used while producing isotopic classifications of loops. We first describe the generation of universal identities (quasigroup isotopy invariants), which builds on a concept introduced by Falconer [6]. We describe how we obtain these invariants using an interplay of model generation and theorem proving. We also present two more types of invariants, which are based on substructures, which – to the best of our knowledge – have not been reported in the literature on loops, hence represent a novel way of characterising loops. These invariants are derived by systematically examining substructures of loops, and are constructed using symbolic computation techniques.

3.1 Universal Identities

Following Falconer [6], from a given loop identity we derive a universal identity. We first define two additional operations \ and / on a pair of elements such that:

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

Note that these operations are well defined because loops are quasigroups. Given a loop identity $w_1 = w_2$, where w_1, w_2 are words of a loop, i.e., combinations of elements of a loop with respect to the loop operation $\cdot$, we can obtain a *derived* or *universal identity* $\overline{w}_1 = \overline{w}_2$ by recursively applying the following transformations:

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

Here y and z are arbitrary elements of a loop, i.e., new universally quantified variables. As an example, consider the two loops below:

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:

$$\forall x . \forall y_1 . \forall y_2 . (x/y_1) \cdot (((x/y_1) \cdot (y_2 \backslash x)) \cdot (y_2 \backslash x)) = ((x/y_1) \cdot (y_2 \backslash x)) \cdot ((x/y_1) \cdot (y_2 \backslash x)).$$

This universal identity was derived from the following loop identity:

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

To generate universal identities for isotopy invariants, we have to first find an equation that holds in some loops, transform it into a derived identity, and subsequently show that this new identity is indeed a loop invariant. In order to get a large number of invariants we employ a process of interleaving model generation and first order theorem proving with intermediate transformations of the respective results. We first systematically generate identities I for which we check whether they are loop identities, by trying to generate a loop of size ≤ 8 that satisfies I using the model generator Mace. All identities for which a loop exists are then transformed into universal identities U as described above. Each U is then passed to at least one first order theorem prover in order to show that it is an isotopy invariant. We employ both Vampire [15] and E [16] for this task. Combined, these show that around 80% of the universal identities are indeed isotopy invariants. Note that for each universal identity, U, we show that it is an invariant under isotopy independently of the size of a loop. We can therefore reuse these universal identities in different classifications. Consequently, we do not have to repeat this step in every classification, but rather perform it offline and collect universal identities in a pool of confirmed isotopy invariants. To date, we have generated $54,000$ confirmed loop identities, which can be translated into universal identities. From these, we have attempted to prove $8,000$ to be isotopy invariants and have succeeded to show this for $6,831$ using both Vampire and E.

During the classification of loops of a particular size, n, we draw on this pool by first filtering them again by using the model generator Finder to generate loops of size n that satisfy the invariant. We then extract those invariants for which at least one loop of order n exists, and we use only these as potential discriminants. Note that the filter discards any invariants which cannot solve any discrimination problem, as no loop of size n satisfies the invariant property. Then, when we need to discriminate between two loops we test whether one the invariants holds for one and not for the other, using DPLLT or CVClite. While the model generation and theorem proving stages are ran in parallel by distributing the problems on a large cluster of processors for both the generation and testing of invariants, finding a discriminant can still take a very long time. Moreover, universal identities are not necessarily sufficient to discriminate between two non-isotopic quasigroups. We therefore explore only a limited number of randomly selected invariants, and if this is not successful, we employ two different methods for generating invariants, which are based on the substructures of quasigroups, as described in Sec. 3.2 and Sec. 3.3.

3.2 Substructure Invariants

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 $\gamma(a \cdot b) = \alpha(a) * \beta(b)$.

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 1. Let $(G, \cdot)$ be a quasigroup of order n, and let i, j, k each be integers such that $1 \leq i, j, k \leq n$. 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 1. *Let $(G, \cdot)$ and $(H, *)$ be isotopic quasigroups of order n, and let i, j, k each be integers such that $1 \leq i, j, k \leq n$. 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 easily from Lemma 1.

Continuing with the notation above, fix an element $(A, B) \in G(i, j, k)$, and for each $g_h \in A \cdot B$, $(1 \leq h \leq k)$, 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 (which we will henceforth refer 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 2. *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 1, 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 2. 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 2. *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 Theorem 1 and Lemma 2.

To generate isotopy invariants based on Theorems 1 and 2, we implemented an algorithm that compares the number of elements in substructures for two quasigroups $(G, \cdot)$ and $(H, *)$. This works iteratively, as follows: for $i = 2, \ldots, n-1$, $j = 2, \ldots, n-1$, and $k = max(i, j), \ldots, n$, if $|G(i, j, k)| \neq |H(i, j, k)|$ then return the invariant, otherwise continue. If all the possible substructures are exhausted without yielding an invariant, we perform a frequency analysis for all the $G(i, j, k)$ and $H(i, j, k)$ until we find a pair $G(i, j, k, F)$ and $H(i, j, k, F')$, such that $F \neq F'$. To keep the formulas resulting for these invariants small, we always prefer an existence argument over the actual comparison of numbers of substructures. That is, we give a preference to invariants, such that either $|G(i, j, k)| = 0$ or $|H(i, j, k)| = 0$.

As an example of such an invariant, consider property P_9 below that expresses that there exists a 2 × 2 substructure that contains exactly 2 distinct elements. Note the use of the unique existence quantifier for variables v_1 and v_2.

$$P_9\colon \quad \exists r_1, r_2 . \exists c_1, c_2 . \exists! v_1, v_2 . r_1 \neq r_2 \wedge c_1 \neq c_2 \wedge v_1 \neq v_2 \wedge \bigwedge_{k=1}^{2} \bigwedge_{j=1}^{2} \bigvee_{k=1}^{2} (r_i \cdot c_j = v_k)$$

With respect to P_9, consider these loops:

L_{21}	0	1	2	3	4	5
0	0	1	2	3	4	5
1	1	0	4	5	2	3
2	2	3	0	1	5	4
3	3	2	5	4	0	1
4	4	5	1	0	3	2
5	5	4	3	2	1	0

Note that L_{21} satisfies property P_9, due to the boxed structure, which is a 2×2 structure with exactly 2 elements. However, P_9 does not hold for L_{23}, i.e., $|L_{23}(2,2,2)| = 0$.

L_{23}	0	1	2	3	4	5
0	2	0	1	5	3	4
1	0	1	2	3	4	5
2	1	2	0	4	5	3
3	4	3	5	0	2	1
4	5	4	3	1	0	2
5	3	5	4	2	1	0

3.3 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 3. *Let $(G, \cdot)$, $(H, *)$, (α, β, γ), i, j, k, n be as in Lemma 2, 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, then $(A', B') \in AB(i', j', k)$ if and only if $(\alpha(A'), \beta(B')) \in \alpha(A)\beta(B)(i', j', k)$. The result now follows in a straightforward manner.

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

Notation 3. 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 3. *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 $|G(i, j, \mathfrak{P}_{i',j'})| = |H(i, j, \mathfrak{P}_{i',j'})|$.*

Proof. This follows immediately from Lemma 3.

In order to generate additional invariants for a given pair of quasigroups $(G, \cdot)$ and $(H, *)$, we employ Theorem 3 in a similar fashion to that used in Sec. 3.2. That is, we successively compare the number of patterns of the same size and the same number of distinct elements.

4 Simplifying Problems

The most difficult verification problems which occur during the classification process involve showing that a given node forms an isotopy class, i.e., we need to prove that a particular set of properties is indeed classifying with respect to isotopy. The opposite problem is to take a given node that doesn't form an isotopy class, and construct a representant that is not isotopic to the existing representant of the node. To solve these two problems with automated theorem provers and model generators requires considerable effort, as basic formalisations of these problems are not solvable using state of the art techniques. We therefore employ several computational methods, implemented in the computer algebra system GAP [8], for reducing the complexity of the problems to be solved. The methods we developed for isotopism problems make use of those we developed for the corresponding isomorphism problems and also exploit some known results from the isotopy theory of quasigroups. Hence, we first give a brief account of how we handled isomorphism problems (for further details see [12]), and then present their adaptation for isotopism, using theorems presented in [14].

4.1 Handling of Isomorphism Problems

In general, to prove that a particular set of properties associated with a node on the decision tree is classifying with respect to isomorphism is a higher-order problem, as it requires proving that all structures satisfying the properties are isomorphic. However, since we are concerned with finite structures, the proof problem can be reduced to first-order and even propositional logic by enumerating all (finitely many) possible isomorphic structures of the representant, Q, of the node. With all isomorphic structures available, it suffices to prove that each structure that satisfies the properties of the node equals one of the isomorphic structures. For the construction of a non-isomorphic structure, it suffices to generate a structure that satisfies the properties of the node but differs from each of the isomorphic structures.

A naive approach to enumerate all isomorphic structures considers all $n!$ possible bijective mappings for structures of cardinality n. From each of these bijective mappings, a structure can be constructed that is isomorphic to Q, and these $n!$ are all possible isomorphic structures of Q. This naive approach can be simplified by the usage of generating systems. A structure Q with binary operation $\circ$ is said to be *generated* by a set of elements $\{q_1, \dots, q_m\} \subseteq Q$ if every element of Q can be expressed as a combination — usually called a factorisation or word — of the q_i under the operation $\circ$. We call a set of generators together with the corresponding factorisations a *generating system*. Given a generating system, we can exploit the fact that each isomorphism is uniquely determined by the images of the generators, to reduce the total number of isomorphisms that we need to consider. If n is the cardinality of the structures and m is the number of generators, then, instead of $n!$, there are only $\frac{n!}{(n-m)!}$ possible mappings and possible resulting structures to consider, since only the m generators have to be mapped explicitly.

In theory, the worst case complexity resulting from the simplification with generating systems is still $n!$, because there is no guarantee that a generating system with a small number of generators exists. In our experiments, however, the number of generators was typically only 2, so the complexity for isomorphism problems was reduced to n^2 from $n!$. This approach requires additional effort in order to compute a (minimal) generating system for Q, as well as to prove that a computed system is indeed a generating system for Q. The former task is performed by GAP, whereas the latter task results in an additional proof problem, which is trivial even for large n.

4.2 Handling of Isotopism Problems

Unfortunately, we cannot directly employ the trick of using generating systems since we now have three, instead of one, bijective mappings to consider, so the images of the generators no longer uniquely determine isotopisms. This means a naive approach to performing the isotopism proof or model construction would have to consider all $(n!)^3$ possible triples of bijective mappings to be applied to a representant Q of a node in the classification tree. As for the case of isomorphism, from the $(n!)^3$ possible triples of bijective mappings, all $(n!)^3$ structures that are isotopic to Q can be computed. With all the isotopes of Q available, an automated theorem prover can be used to show that each structure that satisfies the properties of the node equals one of the isotopes. Moreover, a model generator can be asked to compute a structure that satisfies the properties of the node but differs from each of the isotopes. However, we have found that this naive approach exceeds the abilities of state of the art systems, even for $n = 4$.

In order to simplify our problems, particularly by making use of the reduction offered by generating systems, we exploit the following known results from the isotopy theory of quasigroups (see [14] for details).

Definition 2. *A quasigroup $(G, \cdot)$ is a* principal isotope *of the quasigroup $(G, *)$ if there are permutations α, β, ι on G such that for all $x, y \in G, \alpha(x) * \beta(y) = \iota(x \cdot y)$, where ι is the identity permutation.*

Theorem 4. *If $(G, \cdot)$ and $(H, *)$ are isotopic quasigroups, then $(H, *)$ is isomorphic to some principal isotope of $(G, \cdot)$.*

Definition 3. *Let $(G, \cdot)$ be a quasigroup. Let f and g be fixed elements of G. Then the isotope $(G, *)$ such that $(x \cdot g) * (f \cdot y) = x \cdot y$ for all $x, y \in G$ is called the* fg-isotope *of $(G, \cdot)$.*

Theorem 5. *Let $(G, \cdot)$ and $(H, *)$ be quasigroups. H is isotopic to G if and only if it is isomorphic to an fg-isotope of G.*

These results show that, for our purposes, we no longer need to consider quasigroups with distinct ground sets (the set G will typically suffice). Moreover, it is easy to show that every fg-isotope is, in fact, a loop with unit element fg. Consequently, every quasigroup has a loop isotope, and hence, rather than compute all $(n!)^3$ isotopes for a structure Q, it suffices to:

1. compute the set of all fg-isotopes of Q, $\mathcal{FG}(Q)$,
2. remove structures from $\mathcal{FG}(Q)$ that are isomorphic to other structures in $\mathcal{FG}(Q)$ until the resulting set $\mathcal{FG}'(Q)$ contains no pair of isomorphic structures,

3. compute the set of all structures that are isomorphic to a structure in $\mathcal{FG}'(Q)$. We denote this set: $\mathcal{I}_{fg}(Q)$.

For a given representant, Q, there are n^2 fg-isotopes and for each fg-isotope there are $n!$ isomorphic structures. Hence, our simplification reduces the complexity from $(n!)^3$ structures in the naive approach to $n!n^2$ structures. Although this is already a considerable improvement, the resulting complexity still exceeds the capabilities of state of the art systems, even for small n. We now observe that step 3 discusses isomorphisms only, which consequently allows us to exploit the generating systems simplification developed for the isomorphism problems. Hence, instead of step 3 above, we can perform the following alternative steps:

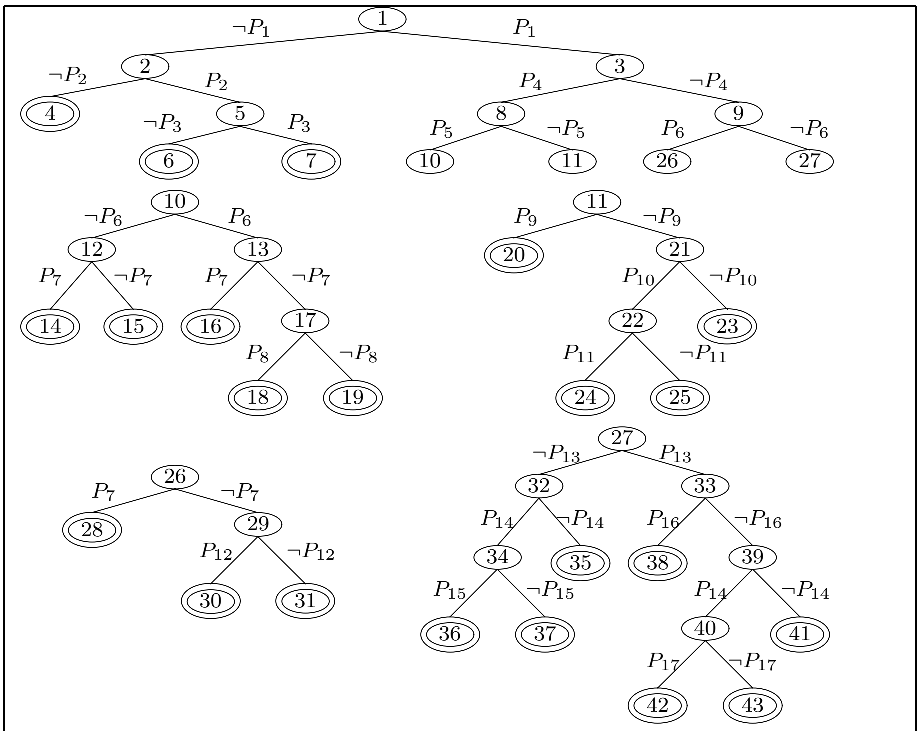
3a. compute a generating system for each structure in $\mathcal{FG}'(Q)$, respectively,
3b. for each pair of a structure in $\mathcal{FG}'(Q)$ and its generating system, compute the set of isomorphic structures with respect to the generating system, respectively. Then, compute the union $\mathcal{G}_{fg}(Q)$ of all resulting structures.

In theory, the worst case complexity of $\mathcal{G}_{fg}(Q)$ is still $n!n^2$. However, in our experiments, we found that the complexity of $\mathcal{FG}'(Q)$ is typically n rather than n^2, and the complexity resulting from the isomorphisms typically comes to n^2 instead of $n!$ when the generating system simplification is exploited. Hence, in practise, the simplified isotopism proof and model generation formalisations typically require the consideration of just n^3 structures. Note that additional effort is necessary to prove that two given structures are isomorphic in step 2, and to compute and verify generating systems in step 3a. However, these additional problems are trivial even for large problems.

5 Results

Our primary result is a new qualitative isotopy classification theorem for loops of order 6, in the sense that it provides a full set of classifiers for the 22 known isotopism classes and corresponding representants. The decision tree for the theorem is given in Fig. 1 (for display purposes, we have broken it up into five parts). The entire tree consists of 43 nodes, where the doubly circled leaf nodes represent isotopy classes. The respective classifying properties correspond to the conjunction of discriminants given along the path from the leaves to the root. The discriminants themselves are given below the tree in Fig. 1.

Note that the top three discriminants are universal identities, while the remainder are substructure invariants, with the exception of P_{11}, which is a pattern invariant. The pattern invariant and four of the substructure invariants (P_4, P_5, P_6, P_9) use an existence argument. The reason for the relatively low number of universal identities as discriminants is that in each step we only test 100 invariants randomly selected out of the overall pool of known invariants before testing for discriminating substructure properties. This restriction is necessary as in our current implementation, testing universal identities with respect to discrimination still consumes a lot of search time. However, we hope that with more advanced encoding techniques and better pre-selection criteria for universal



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: $\forall x, y_1, y_2 \centerdot (x/y_1)\cdot(((x/y_1)\cdot(y_2\backslash y_1))\cdot(y_2\backslash y_1)) = ((x/y_1)\cdot((x/y_1)\cdot(y_2\backslash y_1)))\cdot(y_2\backslash y_1)$

P_3: $\forall x, y_1, y_2 \centerdot (x/y_1) = ((x/y_1)\cdot(y_2\backslash y_1))\cdot(y_2\backslash y_1)$

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}: $|G(4,4,\mathfrak{P}_{2,2})| = 0 \wedge \mathfrak{P}_{2,2} = (0,2,12,23,0,0)$

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}: $|G(2,2,2)| = 11$

Fig. 1. Decision tree and discriminating properties for the loops 6 classification

identity invariants, we will be able to replace more substructure discriminants in the decision tree by universal identities.

For the construction of the models in the tree, we used the Sem, Finder, and DPLLT systems. Apart from the proof problems concerning universal identities, which were carried out by Vampire or E, the majority of the verification proofs have been done with CVClite or DPLLT. A translation into a purely propositional problem without equality, was prohibitive, however, as both the number of variables and nesting depth of the quantifications led to intractably large boolean satisfiability problems. While most properties of the tree are fully automatically verified, we still had difficulty proving that some of the leaf nodes are indeed isotopy classes. However, we hope to finalise this by shifting to the new version of DPLLT in the future. Moreover, given that the number of isotopy classes of order 6 is indeed 22 as known from the literature [11,14] and given the proved discriminating nature of the classifying properties, the decision tree does indeed constitute a classification theorem.

We are currently working on the classification theorem for loops of order 7, for which there are 564 isotopy classes. So far, we have generated 563 nodes in the decision tree, which contains 439 substructures invariants, with an additional 7 employing frequencies, and 117 depending on pattern invariants or frequency on patterns. We suspect that the remaining isotopy class can be discriminated by extending the concept of patterns by a recursion depth. In many cases, the model generation fails to produce any results, so we re-use results from our isomorphism classification of loops of order 7, as well as enumeration results by McKay and Myrwold [11], to search for non-isotopic models. That is, we start with a set of isomorphism class representants of order 7 loops and search amongst those for one that is non-isotopic to a given loop. Once we have finished the tree, we hope to simplify it by searching for useful universal identities.

6 Conclusions and Future Work

We have shown how the bootstrapping algorithm developed in [4] has been extended to produce isotopy classification theorems in loop theory. This involved solving two technical problems, namely the generation of isotopic invariants, and verifying that nodes in the classification tree define isotopy classes. To solve the first problem, we used the notion of universal identities to generate a pool of thousands of invariants, and we also developed new techniques for using concepts based on substructures as invariants. To solve the second problem, we combined facts from algebraic quasigroup theory in a novel computational way to extend techniques we developed in the context of isomorphism classification to isotopy problems. This not only significantly simplified the verification tasks but also made the related task of generating non-isotopic models feasible.

However, it has become clear that it will be difficult to exceed loops of size 7 or 8 with our existing techniques. In particular, the computational techniques to generate substructure invariants need to be improved in order to scale up to larger sizes. This problem can be partially overcome by using more universal identities as discriminants. To this end, it is necessary to cut down on search time by using more intelligent pre-selection of universal identity invariants, as well as producing better reformulations of problems involving universal identities to make them accessible to efficient SAT solving techniques. It is also not clear that our current set of invariants will suffice for all orders. Indeed, we have already investigated another kind of invariant – which extends the concept of frequency to patterns – but which has not been used in the context of the results presented in this paper. Finding new types of isotopy invariants for loops is certainly interesting from a mathematical viewpoint.

We plan to extract details of the bespoke invariant generation techniques developed for isotopy into more extensive background knowledge and improved production rules for the machine learning approach. We believe that it may be possible to re-instate the learning approach, thus restoring a more generic approach to invariant discovery, and possibly discovering new invariants. We also plan to undertake a detailed analysis of the decision trees, as well as a comparison of classification theorems of different sizes for particular structures. From a purely technical point of view, an efficient reimplementation of the entire bootstrapping algorithm would be desirable to overcome some limitations of the Lisp implementation and choice of data structures.

Historically, classification projects have been undertaken for algebras with relatively few classes for small orders (e.g., there are only 5 groups of size 8 up to isomorphism). However, classification of other algebras are no less valid projects, albeit ones which are possibly more suited to an automated approach, due to the large number of classes of small order (e.g., there are $106,228,849$ loops of size 8 up to isomorphism and $1,676,267$ up to isotopism). We have shown that it is possible to make progress on large classification projects, but that this requires the combination of reasoning techniques, including theorem proving, model generation, machine learning, satisfiability solving and symbolic algebra. We believe that such integrated approaches will lead the way in the application of automated reasoning to complex mathematical discovery tasks.

References

1. R. Alur and D. Peled, eds. *Proc. of CAV 2004*, *LNCS* 3114. Springer Verlag, 2004.
2. C. Barrett and S. Berezin. CVC Lite: A New Implementation of the Cooperating Validity Checker. In Alur and Peled [1], pages 515–518.
3. S. Colton. *Automated Theory Formation in Pure Mathematics.* Springer, 2002.
4. S. Colton, A. Meier, V. Sorge, and R. McCasland. Automatic Generation of Classification Theorems for Finite Algebras. In *Proc. of IJCAR 2004*, *LNAI* 3097, pages 400–414. Springer Verlag, 2004.
5. S. Colton and S. Muggleton. Mathematical Applications of Inductive Logic Programming. *Machine Learning Journal*, 2006 (forthcoming).
6. E. Falconer. Isotopy Invariants in Quasigroups. *Transactions of the American Mathematical Society*, 151(2):511–526, 1970.
7. H. Ganzinger, G. Hagen, R. Nieuwenhuis, A. Oliveras, and C. Tinelli. DPLL(T): Fast Decision Procedures. In Alur and Peled [1], pages 175–188.
8. GAP Group. *Groups, Algorithms, and Programming, v4.7*, 2002. `gap-system.org`.
9. W. McCune. Single axioms for groups and Abelian groups with various operations. *Journal of Automated Reasoning*, 10(1):1–13, 1993.
10. W. McCune. *Mace4 Reference Manual and Guide.* Argonne Nat. Laboratory, 2003.
11. B. McKay, A. Meynert, and W. Myrvold. Counting Small Latin Squares. In *Europ. Women in Mathematics Int. Workshop on Groups and Graphs*, pages 67–72, 2002.
12. Andreas Meier and Volker Sorge. Applying SAT Solving in Classification of Finite Algebras. *Journal of Automated Reasoning*, 34(2):34 pages, 2005.
13. M. Moskewicz, C. Madigan, Y. Zhao, L. Zhang, and S. Malik. chaff: Engineering an efficient SAT Solver. In *Proc. of DAC-2001*, pages 530–535, 2001.
14. H. Pflugfelder. *Quasigroups and Loops: Introduction*, volume 7 of *Sigma Series in Pure Mathematics.* Heldermann Verlag, Berlin, Germany, 1990.
15. A. Riazanov and A. Voronkov. Vampire 1.1. In *Proc. of IJCAR 2001*, *LNAI* 2083, pages 376–380. Springer Verlag, 2001.
16. S. Schulz. E: A Brainiac theorem prover. *J. of AI Comm.*, 15(2–3):111–126, 2002.
17. J. Schwenk. A classification of Abelian quasigroups. *Rendiconti di Matematica, Serie VII*, 15:161–172, 1995.

18. J. Slaney. *FINDER, Notes and Guide.* Center for Information Science Research, Australian National University, 1995.
19. J Slaney, M Fujita, and M Stickel. Automated reasoning and exhaustive search: Quasigroup existence problems. *Computers and Mathematics with Applications*, 29:115–132, 1995.
20. G. Sutcliffe. The IJCAR-2004 Automated Theorem Proving Competition. *AI Communications*, 18(1):33–40, 2005.
21. C Weidenbach, U Brahm, T Hillenbrand, E Keen, C Theobald, and D Topic. SPASS Version 2.0. In *Proc. of CADE–18*, *LNAI* 2392, pages 275–279. Springer, 2002.
22. J. Zhang and H. Zhang. *SEM User's Guide.* Dep. of Comp. Science, UIowa, 2001.