Combining AI Systems for Classification in Non-associative Algebra

Volker Sorge[a], Andreas Meier[b], Roy McCasland[c1], Simon Colton[d]

[a] *School of Computer Science, University of Birmingham, UK*
V.Sorge@cs.bham.ac.uk
[b] *DFKI GmbH, Saarbrücken, Germany*
ameier72@web.de
[c] *School of Informatics, University of Edinburgh, UK*
rmccasla@inf.ed.ac.uk
[d] *Department of Computing, Imperial College London, UK*
sgc@doc.ic.ac.uk

Abstract

The automatic construction of mathematical theorems is a challenging task for Artificial Intelligence systems, which has pushed many research boundaries in different branches of AI. We describe how the construction of classification theorems in algebraic domains of mathematics has driven research not only on the individual mathematical reasoning techniques, but also the integration of these techniques. We have developed a bootstrapping algorithm for the automatic generation of such theorems which relies on the power of model generation, first order theorem proving, computer algebra, machine learning and satisfiability solving. In particular, the construction of algebraic invariants demands an intricate interplay of these techniques. We demonstrate the effectiveness of this approach by generating novel theorems which have so far been beyond the reach of automated reasoning systems.

Key words: Automated Discovery, Finite Algebra

1 Introduction

Classification of abstract objects such as integers, graphs, knots and groups has been a major driving force in pure mathematics. Classification projects have been particularly fruitful in the domain of finite algebra, for instance the classification of finite simple groups has been hailed as "one of the greatest intellectual achievements of the 20th Century" [9]. However, the majority of

[1] The author's work was supported by EPSRC MathFIT grant GR/S31099.

finite algebraic structures have yet to succumb to an appropriate classification such as isomorphism or its generalisation, isotopism.

We are interested in finite algebraic structures that consist of one set together with a binary operation on the elements of the set. They are characterised in terms of the axioms to which the multiplication adheres. There has been much research in automated mathematics into finding the most succinct way of representing the axioms of certain algebraic structures. In particular, new axiomatic representations of groups have been found [10,11]. Also, more restrictive axioms bring into question the existence of instances of algebraic structures at particular sizes, and automated techniques have been used to solve such existence problems. By searching for instances of algebraic structures at various sizes, model generators and constraint solvers have proved existence problems both positively, by finding instances, and negatively, by exhausting the search space without finding solutions. The Finder system has been used to solve many quasigroup existence problems [20], and representatives of every isotopy and isomorphism class for quasigroups and loops have been generated up to order 11 [13]. Such enumerations often form an initial part of a classification project.

We are interested in automating more sophisticated classification tasks of algebraic structures with relatively less restrictive axioms. Here the problem is not existence, but rather an abundance of instances, which gives rise to similarly large numbers of isomorphism and isotopism classes for the structure. Because hand-crafting classification theorems with potentially thousands of cases is infeasible, any automated approach to generating such theorems has the potential to substantially add to the body of knowledge in pure mathematics. We present a fully automatic bootstrapping algorithm for constructing classification theorems directly. It is given three inputs: the axioms, A, of a structure of interest, the equivalence relation, E, which defines the classes for the classification, and the size n of the algebraic structure for which the tree is required. The output is a fully verified classification theorem expressed as a decision tree which can decide the equivalence class of any algebraic structure of size n satisfying A. By fully verified, we mean that in addition to the theorem, a proof of its correctness and completeness is also provided.

While the algorithm is general in principle, in practise some additional effort is needed to obtain results. In particular, a key aspect of the algorithm is its ability to generate *algebraic invariants*, which are properties of algebraic structures that are fixed for members of the same equivalence class. Another aspect is that proof problems have to be suitably simplified to be feasible for automated theorem proving. These simplifications often need to exploit knowledge on both the equivalence relation and the algebraic structures.

We have concentrated our efforts on quasigroups and loops together with the two most important algebraic equivalence relations in that domain, namely isomorphism and isotopism, which are defined in the next section. For isomorphism classifications, we used a machine learning approach to generate invari-

ants. However, it turned out that this approach was not sufficient for isotopy classifications, which demanded a significantly more complex approach. We devote the next section to this aspect, before describing the overall bootstrapping algorithm. Following this, we describe the theorems we have generated, most of which are novel and all of which were beyond the reach of existing systems. The original bootstrapping algorithm for isomorphism has been presented in [5]; the mathematical details for generating isotopy invariants are given in [21].

This approach makes use of a dozen different mathematical reasoning implementations which employ first order theorem proving, satisfiability solving, computer algebra, machine learning and model generation. As such, we believe this represents one of the most complex integrations of AI systems yet attempted. We speculate on the future of such combined reasoning systems in the conclusions.

2 Generating Algebraic Invariants

As mentioned above, a key aspect of the bootstrapping algorithm is its ability to generate algebraic invariants. These are properties that are true for every member of an equivalence class for a particular equivalence relation. If, in addition, the property is true for the members of exactly one equivalence class, we call this invariant a discriminant, as it can be used to differentiate between classes. That is, if an algebraic instance has it, and another does not, then they cannot be members of the same equivalence class.

We have concentrated on the domain of quasigroups and loops. Both are structures with one binary operation that have the Latin square property, i.e., each element occurs exactly once in each row and each column of their multiplication table. Loops have, in addition, an identity element [16]. Both quasigroups or loops are generally not associative; indeed an associative quasigroup is a group. We have focused on the two most important equivalence relations in that domain, namely isomorphism and isotopism. We say that two quasigroups $(Q, \cdot)$ and $(Q', \circ)$ are *isomorphic* to each other if there exists a bijective homomorphism f between Q and Q'. That is, $\forall\, x, y \in Q, x = y$ implies $f(x) = f(y)$, and $\forall\, x' \in Q' \exists\, x \in Q$ such that $f(x) = x'$ (bijective). In addition, $\forall x, y \in Q, f(x) \circ f(y) = f(x \cdot y)$ (homomorphism). We call f an *isomorphism* between Q and Q' and the equivalence classes induced by f are termed *isomorphism classes*. We say that two quasigroups $(Q, \cdot)$ and $(Q', \circ)$ are *isotopic* to each other if there are bijective mappings α, β, γ of Q such that $\forall x, y \in Q, \alpha(x) \circ \beta(y) = \gamma(x \cdot y)$. Isotopy is an equivalence relation and the equivalence classes induced by it are called isotopism classes. Isotopism is a generalisation of isomorphism, since Q and Q' are isotopic if $\alpha = \beta = \gamma$. In other words, two structures can be isotopic but not necessarily isomorphic to each other.

As discussed in the next section, the bootstrapping algorithm is required to

generate a discriminant for a given pair of non-equivalent algebraic instances. For isomorphism, we have formulated this as a binary machine learning problem, and used the HR theory formation system to generate suitable discriminants [4,5]. However, this approach was not successful for generating isotopy invariants, and we needed to derive alternative methods, which are described below. Note that due to the fact that every quasigroup has a loop isotope we can restrict the problem of generating such isotopic invariants for loops.

2.1 Isotopy Invariants 1: Substructures

A common mathematical approach to finding invariant properties of algebraic structures is to examine substructures. In contrast to isomorphy, under isotopy columns, rows, and values in the multiplication table can be mapped independently from each other, and a relation between the three can therefore not be exploited to gain an invariant. Nevertheless, we can construct an invariant by looking at the number of different elements a substructure contains. Observe that in our context a substructure is *not* necessarily closed under the operation. The invariants presented here have – to the best of our knowledge – not been reported in the literature on loops, hence represent a novel way of characterising loops. For the mathematical details see [21].

Given a set of row elements $\{r_1, \ldots, r_n\}$ and a set of column elements $\{c_1, \ldots, c_m\}$, we consider their corresponding values in the multiplication table, i.e., elements $v_{11}, \ldots, v_{nm}$ such that $v_{i,j} = r_i \cdot c_j$. We then count the number of distinct elements that make up the v_{ij}. As the γ mapping in the isotopy relation is a bijection, we know that this property is preserved under isotopism. In order to compare this property between loops, however, we have to make it independent from the row and column values. We therefore count the number of substructures of a given size $n \times m$ and the number of distinct values they contain. Thus for every $k \leq n \cdot m$, any isotope of the given loop has to have the same number of $n \times m$ substructures that contain exactly k values. Or, in other words, if two loops differ in the number of $n \times m$ substructures that contain exactly k values, then the two loops are not isotopic.

As an example, consider the following two loops:

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

L_9	0	1	2	3	4	5
0	0	1	2	3	4	5
1	1	0	5	4	2	3
2	2	3	1	5	0	4
3	3	2	4	0	5	1
4	4	5	0	1	3	2
5	5	4	3	2	1	0

Here, L_8 contains 3×3 substructures that contain only three values, for instance the upper left square, whereas L_9 does not contain a single substructure

of that form. Formally we get the property

$$\exists r_1, r_2, r_3. \ \exists c_1, c_2, c_3. \exists! v_1, v_2, v_3. \\ r_1 \neq r_2 \wedge r_1 \neq r_3 \wedge r_2 \neq r_3 \wedge c_1 \neq c_2 \wedge c_1 \neq c_3 \wedge c_2 \neq c_3 \\ \wedge v_1 \neq v_2 \wedge v_1 \neq v_3 \wedge v_2 \neq v_3 \wedge \bigwedge_{i=1}^{3} \bigwedge_{j=1}^{3} \bigvee_{k=1}^{3} (r_i \cdot c_j = v_k)$$

Note that the above formulation of the discriminant contains a unique existence quantifier $\exists!$. Expanding this would lead to an even larger formula. Nevertheless, the above property is still relatively small. Generally we have to quantify the different substructures in the loops under consideration to obtain a discriminant. That is, we have to formulate the concept that one loop contains more substructures of a certain type than the other one.

2.2 Isotopy Invariants 2: Universal Identities

A second type of isotopy invariant involves the concept of universal identities, first introduced be Falconer [6]. Before defining universal identities, we have to introduce the general concept of loop identities.

A loop identity is a universally quantified equation of the form $w_1 = w_2$, where w_1, w_2 are words of a loop, i.e., combinations of variables and the loop operation $\cdot$. Every loop identity holds for the trivial loop, i.e., the loop with only one element, however many identities (e.g., $\forall x, y. x \cdot y = x$) do not hold for any loop of greater order. Moreover, while identities are preserved under isomorphisms, they are not necessarily preserved under isotopism.

In [6], Falconer introduced the idea of *universal identity*, namely one that is preserved under isotopy. Universal identities depend on the concept of *derived identities* that is defined using two additional quasigroup operations / and \, which, in turn, can be defined as

(i) $x \cdot (x/y) = y$ and $x/(x \cdot y) = y$,
(ii) $(x\backslash y) \cdot x = y$ and $(y \cdot x)\backslash x = y$,

by exploiting the fact that the operation of a quasigroup is left and right cancellative. Observe that we use / and \ similar to Falconer in [6], while the symbols are often used conversely by other authors (cf. [15,3]). Given a universally quantified identity $w_1 = w_2$ where w_1 and w_2 are words in n variables $x_1, \ldots, x_n$ and the operation $\cdot$ we create an identity $\overline{w_1} = \overline{w_2}$ in $n+2$ variables $x_1, \ldots, x_n, y, z$ and the operations $(\cdot, \backslash, /)$ where $\overline{w}$ is defined recursively as follows

(i) $\overline{x_i} = x_i$
(ii) $\overline{u \cdot v} = (\overline{u}\backslash y) \cdot (z/\overline{v})$

The identity $\overline{w_1} = \overline{w_2}$ is called a *derived identity*, the definition of $\overline{\ }$ given here is a simplification of that given in [6]. Falconer shows that a derived identity, that is constructed from a loop identity that holds for the free loop, is always

a universal identity, that is an isotopism invariant.

As an example consider the two loops and their isotopy discriminant below. The given universal identity holds for L_4 but does not hold for L_8.

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

$$\forall x_{\blacksquare}\forall y_{\blacksquare}\forall z_{\blacksquare}(x/y)\cdot(((x/y)\cdot(z\backslash x))\cdot(z\backslash x))=((x/y)\cdot(z\backslash x))\cdot((x/y)\cdot(z\backslash x))$$

This universal identity was derived from the following loop identity:

$$\forall x_{\blacksquare}x\cdot((x\cdot x)\cdot x)=(x\cdot x)\cdot(x\cdot x)$$

Falconer's concept of universal identity depends on deriving identities from equations that hold for the free loop. However, in our scenario we can only generate finite models and a directed search for identities holding for the free loop is infeasible. Thus to generate a large number of isotopy invariants we employ a process of interleaving model generation and first order theorem proving with intermediate transformations of the respective results. Figure 1 schematically depicts this process. We first systematically generate identities I for which we check whether they are loop identities, by trying to generate a non-trivial model of size $4 \leq n \leq 8$ that satisfies I using the model generator Mace [12]. All identities for which a loop exists are then transformed into derived identities U as described above. Each U is then passed on to at least one first order theorem prover in order to show that it is indeed an isotopy invariant, which is not necessarily guaranteed anymore by construction. We employ both Vampire [17] and E [18] for this task. Combined, these show that around 80% of our derived identities are indeed isotopy invariants. These identities are then collected in a pool of confirmed isotopy invariants.

Note that for each 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. If we need to test some isotopy invariants if they can be used as discriminants during the classification of loops of a particular size n, we filter them again using the model generator Finder [19] to generate loops of size n that satisfy the invariant. We then only use those invariants for which at least one loop of order n exists, and try them as possible discriminant during the classification. Note that we use different model generators because Mace can deal better with generating models of different small sizes, whereas Finder is more efficient when faced with more than one operation on a structure. Note also that the model generation and theorem proving stages are run in parallel by distributing the problems on a large cluster.

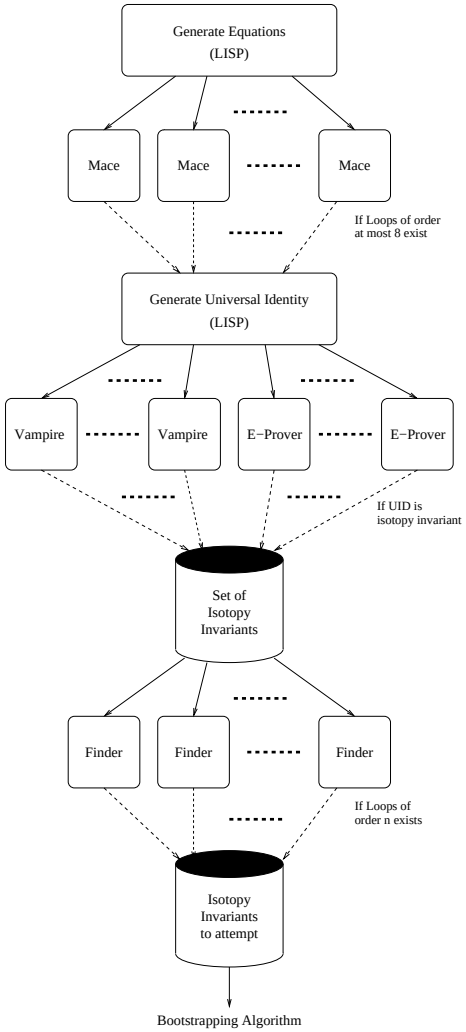

Fig. 1. Generating isotopy invariants from universal identities.

3 Bootstrapping Algorithm

The bootstrapping algorithm to generate classification theorems 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 full proof of both the correctness and the completeness of the classification theorem is generated. We first give an informal overview of the algorithm before elaborating some of its details more formally in the remainder of the section.

Informally the algorithm works as follows. Firstly, it initialises a decision tree with the 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 works iteratively, starting with the root node and moving on to successive nodes in the tree. The algorithm 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, the algorithm does one of the following two things: (1) It 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) it 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. In 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 structure. This property 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 constructs 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. The leaf nodes then either represent equivalence classes or are empty, i.e., no structure exists with the properties given in the node. The final classification theorem corresponds to the disjunction of the properties given as labels of the leaf nodes. An example of a partial decision tree and some details of how it was constructed is given in the results section.

The bootstrapping algorithm itself is a framework that combines several reasoning techniques, which play their part in achieving the overall goal. It relies on other third party systems to generate algebraic structures and discriminants, and to verify the construction of the decision tree in each step. We use the following methodologies in the different steps of the algorithm:

Generating Structures

We use model generation to construct algebraic structures. In the first step, the algorithm simply calls a model generator to return a structure corresponding to the input axioms. Throughout the process, model generators are used to construct example structures that are not equivalent to an already given one. We use three different model generators for this, namely Mace [12], Sem [23], and Finder [19], as each has its particular strengths and weaknesses depending on the problem domain and equivalence relation in question.

Generating Discriminants

The approach to constructing discriminating properties varies from equivalence relation to equivalence relation. We described various approaches to this in the previous section.

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 algebraic structures Q and Q', we show that (1) P is a proper discriminant for the equivalence relation E, (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; we have experimented with several provers. We found the Spass [22], Vampire [17] and E [18] provers to be the most useful for our problems.

Verifying Equivalence Classes

The most difficult verification problems occurring during the classification involve showing that a given node forms an equivalence class with respect to the relation E, i.e., to prove that a particular set of properties is indeed classifying. These types of proof are necessary to fully verify the completeness of a decision tree. More precisely, we have to show that all algebraic structures of cardinality n that satisfy the describing property of the node are equivalent. Although the theorems are essentially second order, since 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. This enables us to translate our problems into satisfiability problems and exploit the strength of state of the art SAT solvers. Depending on the problem domain and equivalence relation in question, we can either use solvers with or without equality. Currently we have integrated the systems zChaff [14], CVClite [2], and DPLLT [7].

If translated naively, many of the proof problems would be beyond the capabilities of existing SAT solvers or theorem provers. To decrease complexity we have implemented some computer algebra algorithms in GAP [8] that exploit some mathematical domain knowledge to reduce the complexity of the isomorphism and isotopism problems we have encountered. In particular, we have implemented the following methods:

Verifying Isomorphism Classes

For the isomorphism equivalence relation, the number of mappings to consider in a naive approach would be $n!$ for structures of size n. However we can improve upon this by exploiting the notion of a generating system: a structure A with binary operation $\circ$ is said to be *generated* by a set of elements $\{a_1, \ldots, a_m\} \subseteq A$ if every element of A can be expressed as a combination – usually called a factorisation or word – of the a_i under the operation $\circ$. We call a set of generators together with the corresponding factorisations a *generating system*. Generating systems are invariant under isomorphism. Hence, if we can show that all algebraic structures with cardinality n satisfying a property P share a generating system, we can reduce the number of mappings to consider, when showing that all structures satisfying P are isomorphic, to the mappings on generators only.

Verifying Isotopism Classes

For the isotopism equivalence relation, the naive approach is even worse than for isomorphism, since we would need to consider $(n!)^3$ mappings. We cannot cut down on the number of cases as easily as in the isomorphism case by using generating systems since they are not invariant under isotopy. Instead, our simplification strategy depends on the fact that a loop Q is an isotope of Q' if it is isomorphic to an *fg-isotope* of Q', where $(Q', *)$ is an fg-isotope of $(Q', \cdot)$ if $(x \cdot g) * (f \cdot y) = x \cdot y$ for all $x, y \in Q'$ and some fixed elements $f, g \in Q'$ (i.e., $*$ operates on the same set of elements as $\cdot$) [16]. Now suppose we want to show that a node with representant Q and properties P is indeed an isotopism class. We first compute all fg-isotopes of Q that are not isomorphic to each other together with their generating systems. We then show that all algebraic structures with cardinality n satisfying a property P have at least one of the generating systems of the fg-isotopes. This particular problem can amount to rather large disjunctive formulas. If we can show this, then we can reduce the proof that all structures in our isotopism class are isotopic to Q to the statement that all structures in the class are isomorphic to one of the fg-isotopes. In the formulation of this problem we can now exploit the fact that we have the generating systems for all fg-isotopes and therefore only have to show all possible mappings to the different generating systems involved.

In addition to exploiting the above techniques for the verification proofs we can apply them when looking for non-isomorphic or non-isotopic models, respectively, during the bootstrapping algorithm.

4 Results

We have experimented with the bootstrapping algorithm to classify several types of algebraic structures with respect to isomorphism and isotopism. The results are given in Table 1. The left column of the table gives the domain of the classification, i.e., quasigroups or loops together with the order of the structures and possibly additional axioms.

For example, *Quasigr.6 + Idem.* denotes the domain of quasigroups of order 6 together with idempotency as an additional axiom. Observe that for some domains, the classification is not yet completed, mostly because the process is very time and computation intensive. The remaining three columns contain information about the form of the decision trees constructed with the bootstrapping algorithm, such as the total number of nodes in the tree, the number of equivalence classes, i.e., leaf nodes of the tree, and the maximum depth of the tree. The largest decision tree so far is the full isomorphism classification of quasigroups of size 5 that contains 2875 nodes and 1411 isomorphism classes but is nevertheless fairly shallow with a maximum depth of 23. Its completion alone has taken more than four month. In addition to classifications of quasigroups and loops we have also tested the algorithm

Algebraic Structure	Nodes	Equiv.Classes	Depth
Isomorphism Classification			
Quasigroup 3	9	5	4
Quasigroup 4	71	35	9
Quasigroup 5	2875	1411	23
Quasigroup 6 + Idem.	35	18	6
Quasigroup 6 + QG3*	131	64	11
Quasigroup 7 + QG9*	397	108	12
Loops4	3	2	2
Loops5	11	6	5
Loops6	233	109	17
Loops7*	109	53	13
Isotopism Classification			
Loops5	3	2	1
Loops6	43	22	8

* not yet completed.

Table 1
Generated classification theorems.

with classifications of monoids, groups and star algebras with respect to isomorphism [5].

As an illustrative example, consider the isotopy classification of loops of size 6. The top of the tree produced by the bootstrapping algorithm is given in Figure 2. Single circled nodes represent inner nodes of the tree and doubly circled nodes represent leaf nodes, which correspond to isotopy classes. For clarity we have written the discriminating properties as labels on the edges, thus the property for a given node can be obtained by collecting the properties along the path from the node to the root of the tree. The discriminating properties $P_1, \ldots, P_4$ are given below the tree.

When we start the algorithm, the first loop generated is L_4 given in the previous section (it is labelled L_4 because the loop will eventually become the representant of the isotopy class given by node 4 in the tree). In the next step, the algorithm tries to find another loop that is not an isotope of L_4. It succeeds and produces the structure L_8, which will eventually become the representant of node 8 and is likewise given above. The algorithm then finds the universal identity P_1 to discriminate L_4 and L_8, where P_1 holds for L_4 but not for L_8. This leads to the addition of two new nodes 2 and 3.

In the next bootstrapping round, node 2 is further explored. Again, a loop is found that is non-isotopic to L_4 but that nevertheless satisfies P_1, which

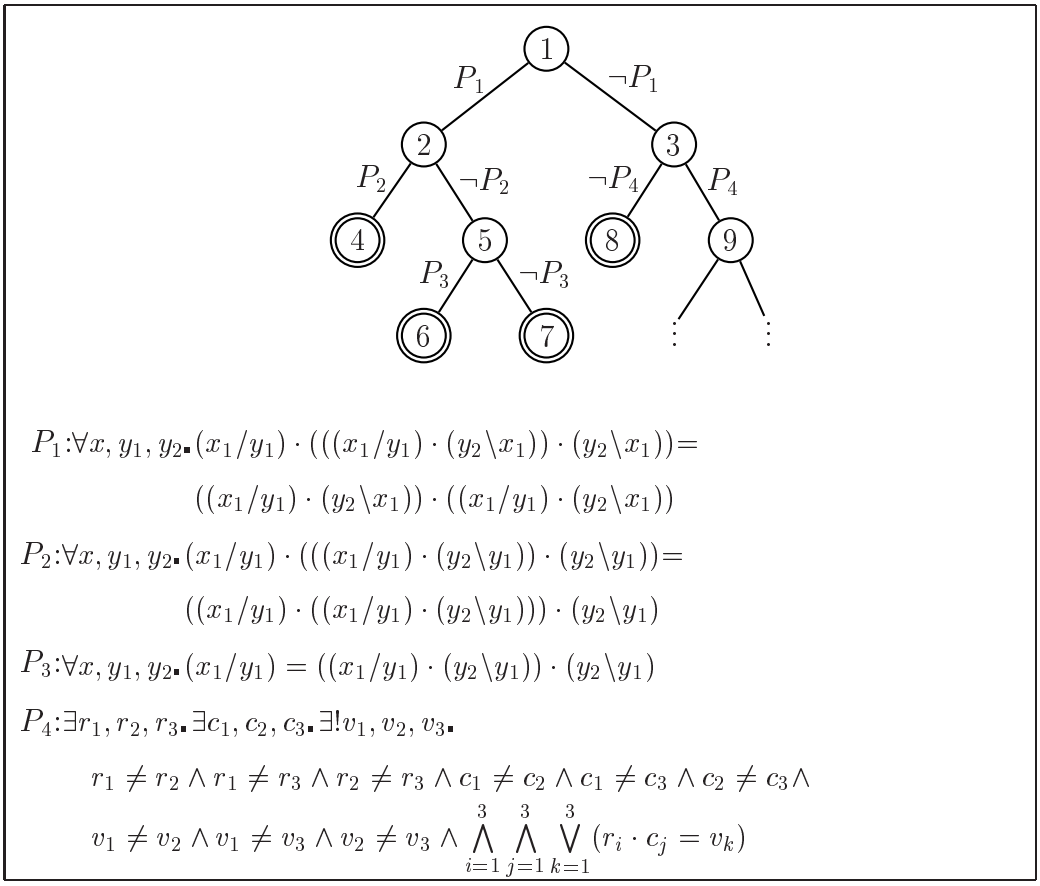

Fig. 2. Decision tree for isotopy classification of size 6 loops.

subsequently leads to another branching of the tree with P_2 as discriminant. The next node to be expanded is then 4. This time, the model generation fails and no structure can be found that is both not an isotope of L_4 and satisfies P_1 as well as P_2. This suggests that node 4 represents an equivalence class with respect to isotopism, with representant L_4 and $P_1 \wedge P_2$ as the classifying property. This is subsequently shown by a SAT solver. The bootstrapping algorithm then carries on with the expansion of node 5, etc., until the entire tree is explored.

We have omitted showing the entire classification tree, which contains another 34 nodes below node 9. While the top of the classification tree contains universal identities as discriminants, the nodes below node 9 are discriminated by substructure properties only. Moreover, the automatically generated subtree is degenerated, that is, in every branching node has exactly one leaf node and one branching node as children. Manual rotation gives a more balanced tree below node 9, see [21] for details. So far we use roughly 6800 isotopy invariants for our classification but we only explore a limited number of randomly selected invariants before we accept a possible substructure invariant as discriminant.

5 Conclusions and Further Work

We have presented an approach to constructing classification theorems in finite algebra in a bootstrapping framework that facilitates the collaboration of a significant number of AI systems. In our experiments we have focused on the important equivalence relations of isomorphism and isotopism in the algebraic domain of quasigroups and loops. Most of the theorems that our approach has produced so far are certainly novel to mathematics and are most likely difficult to derive by hand. The large number of instances already for relatively small cardinalities makes it necessary and fruitful to develop sophisticated automated techniques for classification. While our approach is general for the isomorphism equivalence relation, i.e. it does not depend on the particular domain of quasigroups and loops, the techniques developed for isotopism depend very much on domain knowledge. In particular, the generation of invariants as well as the simplification necessary to verify isotopism classes exploit the mathematics of the domain of loops. Nevertheless, we consider the non-trivial generalisation of the original techniques to isotopism a significant achievement. And considering the applications of quasigroups, such as in scheduling, code theory, or cryptography, the techniques developed for isotopism proofs might have impact beyond their use in pure classification.

Currently it seems that with structures of order 8 we reach the solvability horizon of most provers and SAT solvers, with our current encoding techniques. In particular, the encoding of the substructure invariants is often too large and cumbersome to get the proofs through. More advanced encoding techniques are therefore subject of future work. Similarly a smarter pre-selection of universal identity invariants would be desirable to cut down on search time and on the need to use substructure invariants. In this context we also intend to have another look at the isotopy classification of loops of size 6, in order to gain a shallower tree. To make further use of the constructed classification theorems we envisage a detailed, possibly automated, analysis of the decision trees and the occurring properties as well as a comparison of classification theorems of different sizes for particular structures. Another approach to get more uniform trees could be to first classify loops with respect to isotopy, then grow this tree further to classify all loops with respect to isomorphism, and afterwards grow the tree even further by classifying all quasigroups of that size.

It is clear that it is impossible to reproduce the classification theorems generated by this approach using a single reasoning system. This highlights the fact that combining reasoning systems can often produce an implementation which is more than the sum of its parts. We believe that only via such integration projects will automated mathematics begin to make an impact on mainstream mathematical research.

References

[1] Alur, R. and D. Peled, editors, "Computer Aided Verification, 16^{th} International Conference, CAV 2004," LNCS **3114**, Springer Verlag, Boston, MA, USA, 2004.

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

[3] Burris, S. and H. P. Sankappanavar, "A Course in Universal Algebra," Graduate Texts in Mathematics **78**, Springer-Verlag, New York, 1981, xvi+276 pp.

[4] Colton, S., "Automated Theory Formation in Pure Mathematics," Springer, 2002.

[5] Colton, S., A. Meier, V. Sorge and R. McCasland, *Automatic generation of classification theorems for finite algebras*, in: *Proc. of IJCAR 2004*, LNAI **3097** (2004), pp. 400–414.

[6] Falconer, E., *Isotopy invariants in quasigroups*, Transactions of the American Mathematical Society **151** (1970), pp. 511–526.

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

[8] The GAP Group, "GAP – Groups, Algorithms, and Programming, Version 4.3," (2002), http://www.gap-system.org.

[9] Humphreys, J., "A Course in Group Theory," Oxford University Press, 1996.

[10] Kunen, K., *Single Axioms for Groups*, Journal of Automated Reasoning **9** (1992), pp. 291–308.

[11] McCune, W., *Single axioms for groups and Abelian groups with various operations*, Journal of Automated Reasoning **10(1)** (1993), pp. 1–13.

[12] McCune, W., "Mace4 Reference Manual and Guide," Argonne National Laboratory (2003), aNL/MCS-TM-264.

[13] McKay, B., A. Meinert and W. Myrvold, *Counting small latin squares* (2003), available at http://www.csr.uvic.ca/~wendym/ls.ps.

[14] Moskewicz, M., C. Madigan, Y. Zhao, L. Zhang and S. Malik, *chaff: Engineering an efficient SAT solver*, in: *Proc. of the 39^{th} Design Automation Conference*, 2001, pp. 530–535. URL http://www.sigda.org/Archives/ProceedingArchives/Dac/Dac2001/

[15] Nagy, G. P. and P. Vojtěchovský, "Computing with quasigroups and loops in GAP," http://www.math.du.edu/loops/.

[16] Pflugfelder, H. O., "Quasigroups and Loops: Introduction," Sigma Series in Pure Mathematics **7**, Heldermann Verlag, Berlin, Germany, 1990.

[17] Riazanov, A. and A. Voronkov, *Vampire 1.1*, in: *Proc. of IJCAR 2001*, LNAI **2083** (2001), pp. 376–380.

[18] Schulz, S., *E: A brainiac theorem prover*, Journal of AI Communication **15** (2002), pp. 111–126.

[19] Slaney, J., "FINDER, Notes and Guide," Center for Information Science Research, Australian National University (1995).

[20] Slaney, J., M. Fujita and M. Stickel, *Automated reasoning and exhaustive search: Quasigroup existence problems*, Computers and Mathematics with Applications **29** (1995), pp. 115–132.

[21] Sorge, V., A. Meier, R. McCasland and S. Colton, *Automatic construction and verification of isotopy invariants*, in: *Proc. of IJCAR 2006*, LNAI **4130** (2006), pp. 36–51, forthcoming.

[22] Weidenbach, C., U. Brahm, T. Hillenbrand, E. Keen, C. Theobald and D. Topic, *SPASS version 2.0*, in: A. Voronkov, editor, *Proc. of the 18th International Conference on Automated Deduction (CADE–18)*, LNAI **2392** (2002), pp. 275–279.

[23] Zhang, J. and H. Zhang, "SEM User's Guide," Department of Computer Science, University of Iowa (2001).