Employing Theory Formation to Guide Proof Planning

Andreas Meier[1], Volker Sorge[2], and Simon Colton[3]*

[1] Fachbereich Informatik, Universität des Saarlandes, Germany,
ameier@ags.uni-sb.de, http://www.ags.uni-sb.de/~ameier
[2] School of Computer Science, University of Birmingham, UK,
V.Sorge@cs.bham.ac.uk, http://www.cs.bham.ac.uk/~vxs,
[3] Division of Informatics, University of Edinburgh, UK,
simonco@dai.ed.ac.uk, http://www.dai.ed.ac.uk/~simonco

Abstract. The invention of suitable concepts to characterise mathematical structures is one of the most challenging tasks for both human mathematicians and automated theorem provers alike. We present an approach where automatic concept formation is used to guide non-isomorphism proofs in the residue class domain. The main idea behind the proof is to automatically identify discriminants for two given structures to show that they are not isomorphic. Suitable discriminants are generated by a theory formation system; the overall proof is constructed by a proof planner with the additional support of traditional automated theorem provers and a computer algebra system.

1 Introduction

In [10] and [11] we present a case study concerned with the automatic classification of residue class sets over the integers into sets of isomorphic structures. The residue class sets have given binary operations and our approach is to prove that two given structures are isomorphic or not in terms of their basic algebraic properties. The necessary proofs are constructed with the MULTI proof planner [13], which forms part of the ΩMEGA theorem proving environment [1]. MULTI uses the guidance of computer algebra and model generation. Particularly hard problems arising in the domain are non-isomorphism proofs, i.e. to show that two given structures are non-isomorphic, since a naïve proof attempt with an exhaustive case analysis is infeasible in most cases. A better approach is to find an *invariant* of the residue class sets (a property which does not change under isomorphism) and show that it differs for a particular pair of residue class sets. We say that the particular invariant found for two structures acts as a *discriminant* for the structures if it has a different value for each. Unfortunately, employing a small set of predefined invariants cannot necessarily deal with all cases which might occur.

* The author's work is supported by EPSRC grant GR/M98012 and European Union IHP grant CALCULEMUS HPRN-CT-2000-00102. He is also affiliated with the Department of Computer Science at the University of York.

In this paper we overcome this dilemma by using the HR automatic theory formation system [3] to automatically detect discriminants for given structures. We then model the discriminants into appropriate concepts and the overall proof is constructed by MULTI with the support of traditional automated theorem provers and a computer algebra system.

The paper is structured as follows: in section 2 we introduce the problem domain and motivate the use of automatic concept formation for non-isomorphism proofs. Section 3 gives a brief overview of the various systems we employ to construct non-isomorphism proofs. Section 4 gives a detailed account of how the proofs are constructed and we present some preliminary results in section 5.

2 Proving Non-Isomorphism Problems

In this section we first briefly introduce the problem domain of residue class structures and the proof techniques we have already developed for non-isomorphism proofs as described in [10, 11]. Following this, we generalise one of these proof techniques to a new proof scheme which enables the construction of arbitrary discriminants for two algebraic structures that suffice to demonstrate that the structures involved are not isomorphic.

2.1 The Residue Class Domain

We define a residue class set over the integers as the set of all congruence classes modulo an integer n, i.e., $\mathbb{Z}_n$, or as an arbitrary subset of $\mathbb{Z}_n$. More concretely, we are dealing with sets of the form $\mathbb{Z}_3, \mathbb{Z}_5, \mathbb{Z}_3\backslash\{\bar{1}_3\}, \mathbb{Z}_5\backslash\{\bar{0}_5\}$, $\{\bar{1}_6, \bar{3}_6, \bar{5}_6\}$, etc. where $\bar{1}_6$ denotes the congruence class of 1 modulo 6. If c is an integer we also write $cl_n(c)$ for the congruence class of c modulo n. A binary operation $\circ$ on a residue class set is given in λ-function notation, and $\circ$ can be of the form $\lambda xy.x$, $\lambda xy.y$, $\lambda xy.c$ where c is a constant congruence class (e.g., $\bar{1}_3$), $\lambda xy.x\bar{+}y$, $\lambda xy.x\bar{*}y$, $\lambda xy.x\bar{-}y$, where $\bar{+}$, $\bar{*}$, $\bar{-}$ denote addition, multiplication, and subtraction of congruence classes over the integers, respectively. Furthermore, $\circ$ can be any combination of the basic operations with respect to a common modulo factor, e.g., $\lambda xy.(x\bar{+}\bar{1}_3)\bar{-}(y\bar{+}\bar{2}_3)$. We often abbreviate the operations $\lambda xy.x\bar{+}y$, $\lambda xy.x\bar{*}y$ and $\lambda xy.x\bar{-}y$ by $\bar{+}$, $\bar{*}$ and $\bar{-}$, respectively.

For two given structures $(RS_n^1, \circ^1)$ and $(RS_m^2, \circ^2)$ we examine whether or not they are isomorphic; that is, we determine whether or not there is a function $h{:}(RS_n^1, \circ^1) \rightarrow (RS_m^2, \circ^2)$ such that h is injective, surjective, and homomorphic.[1] For proof planning both isomorphism and non-isomorphism proofs, the appropriate guidance for the proof planner is crucial for success. In cases where two structures are isomorphic, it is usually fairly simple to compute an appropriate mapping h with either a computer algebra system or a model generator, and

[1] Observe that we avoid confusion between indices and modulo factors by writing indices as superscripts, except in indexed variables such as x_i, y_j as they are clearly distinct from congruence classes of the form $cl_i(x)$.

subsequently show that h is indeed an isomorphism. However, when the structures are not isomorphic, it is much more difficult to appropriately guide the necessary non-isomorphism proof.

2.2 Techniques for Non-Isomorphism Proofs

In our previous work [10, 11], we have implemented several proof techniques for the proof planner MULTI to show that two structures are not isomorphic. These require varying degrees of guidance from computer algebra or model generation:

Testing all possible functions h. Essentially this corresponds to a case split on all possible instantiations for the mapping h and showing in each case that h is not an isomorphism. While this technique does not require any guidance for MULTI, for two structures whose sets have cardinality n, MULTI has to consider n^n possible functions, which becomes infeasible even for relatively small n.

Proof by contradiction. The idea of this technique is to find a pair of distinct elements in one structure that is always mapped to the same image under each homomorphism h. This shows that there exists no injective h and therefore no isomorphism. For this technique, a prospective pair of elements can be computed either with the computer algebra system MAPLE [16] or, more reliably, with the SEM model generator [17]. However, even with this guidance, the subsequent proof process is essentially equational theorem proving, and success is not guaranteed.

Using predefined invariants. An intuitive way to show non-isomorphism is to find an invariant property of one structure that the other structure does not exhibit. We have already implemented a proof planning approach for the following predefined invariants: (1) the structures involved are of different cardinality; (2) the structures form different algebraic entities; e.g., one structure is a group while the other is a semigroup; (3) one of the structures contains an element of some order k and no element in the other structure has order k. For structures without a unit element, we can similarly use the order of traces of elements. MULTI checks these invariants in this order. To compute both orders and traces of elements, MULTI uses the computer algebra system GAP [8]. In the automatic exploration of the residue class domain (see [11]) we usually start with sets of similar algebraic structures of the same cardinality (e.g., quasigroups of order 5). Hence invariant (3) is the only one of relevance, and the predefined criteria are often not sufficient to successfully construct a non-isomorphism proof.

2.3 Systematically Constructing Discriminants

The new proof technique we describe in this paper aims to be a more reliable proof strategy for non-isomorphism proofs. It is essentially a generalisation of the technique presented in section 2.2. Given two structures, we construct an appropriate, bespoke discriminant (i.e., an invariant property that only one of the structures exhibits) to show that the structures are not isomorphic. More

$S^1 = (\mathbb{Z}_5, \bar{-})$

S^1	$\bar{0}_5$	$\bar{1}_5$	$\bar{2}_5$	$\bar{3}_5$	$\bar{4}_5$
$\bar{0}_5$	$\bar{0}_5$	$\bar{4}_5$	$\bar{3}_5$	$\bar{2}_5$	$\bar{1}_5$
$\bar{1}_5$	$\bar{1}_5$	$\bar{0}_5$	$\bar{4}_5$	$\bar{3}_5$	$\bar{2}_5$
$\bar{2}_5$	$\bar{2}_5$	$\bar{1}_5$	$\bar{0}_5$	$\bar{4}_5$	$\bar{3}_5$
$\bar{3}_5$	$\bar{3}_5$	$\bar{2}_5$	$\bar{1}_5$	$\bar{0}_5$	$\bar{4}_5$
$\bar{4}_5$	$\bar{4}_5$	$\bar{3}_5$	$\bar{2}_5$	$\bar{1}_5$	$\bar{0}_5$

$S^2 = (\mathbb{Z}_5, \lambda x y.(\bar{2}_5 \bar{*} x) \bar{+} y)$

S^2	$\bar{0}_5$	$\bar{1}_5$	$\bar{2}_5$	$\bar{3}_5$	$\bar{4}_5$
$\bar{0}_5$	$\bar{0}_5$	$\bar{1}_5$	$\bar{2}_5$	$\bar{3}_5$	$\bar{4}_5$
$\bar{1}_5$	$\bar{2}_5$	$\bar{3}_5$	$\bar{4}_5$	$\bar{0}_5$	$\bar{1}_5$
$\bar{2}_5$	$\bar{4}_5$	$\bar{0}_5$	$\bar{1}_5$	$\bar{2}_5$	$\bar{3}_5$
$\bar{3}_5$	$\bar{1}_5$	$\bar{2}_5$	$\bar{3}_5$	$\bar{4}_5$	$\bar{0}_5$
$\bar{4}_5$	$\bar{3}_5$	$\bar{4}_5$	$\bar{0}_5$	$\bar{1}_5$	$\bar{2}_5$

$S^3 = (\mathbb{Z}_5, \lambda x y.(\bar{3}_5 \bar{*} x) \bar{+} y)$

S^3	$\bar{0}_5$	$\bar{1}_5$	$\bar{2}_5$	$\bar{3}_5$	$\bar{4}_5$
$\bar{0}_5$	$\bar{0}_5$	$\bar{1}_5$	$\bar{2}_5$	$\bar{3}_5$	$\bar{4}_5$
$\bar{1}_5$	$\bar{3}_5$	$\bar{4}_5$	$\bar{0}_5$	$\bar{1}_5$	$\bar{2}_5$
$\bar{2}_5$	$\bar{1}_5$	$\bar{2}_5$	$\bar{3}_5$	$\bar{4}_5$	$\bar{0}_5$
$\bar{3}_5$	$\bar{4}_5$	$\bar{0}_5$	$\bar{1}_5$	$\bar{2}_5$	$\bar{3}_5$
$\bar{4}_5$	$\bar{2}_5$	$\bar{3}_5$	$\bar{4}_5$	$\bar{0}_5$	$\bar{1}_5$

Fig. 1. Some quasigroup multiplication tables

formally, for two structures S^1 and S^2 we want to find a property P such that $P(S^1) \wedge \neg P(S^2)$ holds[2].

For example, consider the pairwise non-isomorphic quasigroups S^1, S^2, S^3 given with their respective multiplication tables in Fig. 1. When comparing the tables of S^1 and S^2, one discriminant is fairly obvious: while S^1 has only $\bar{0}_5$ on the main diagonal, all elements on the main diagonal of S^2 are distinct. Thus, the invariant property we can use is $\exists x. \forall y. x = y \circ y$. Things become less obvious when we compare the multiplication tables of S^2 and S^3. Here, one invariant of S^3, which does not hold for S^2, is $\forall x. \forall y. (x \circ x = y) \Rightarrow (y \circ y = x)$,.

The generalised proof procedure is as follows: given two structures S^1 and S^2 we have to:

1. find an appropriate discriminant P,
2. show that $P(S^1)$ holds,
3. show that $\neg P(S^2)$ holds, and finally
4. show that $\forall X. \forall Y. P(X) \wedge \neg P(Y) \Rightarrow X \not\sim Y$ holds [3].

The single proof parts combine to give the following, sketched formal proof:

$$\dfrac{\dfrac{\begin{array}{c}?\\ \vdots\ (2)\end{array} \quad \dfrac{P(S_1) \quad \neg P(S_2)}{P(S_1) \wedge \neg P(S_2)}\wedge Intro \quad \dfrac{\begin{array}{c}?\\ \vdots\ (3)\end{array}}{\forall X. \forall Y. P(X) \wedge \neg P(Y) \Rightarrow X \not\sim Y} \quad \dfrac{\begin{array}{c}?\\ \vdots\ (4)\end{array}}{P(S_1) \wedge \neg P(S_2) \Rightarrow S_1 \not\sim S_2}\forall Elim(S_1, S_2)}{S_1 \not\sim S_2} ModusPonens$$

In the following section, we describe how we realize this proof technique with a combination of MULTI and several other heterogenous systems. As we observed in the example, discriminants are not necessarily obvious and can have fairly complicated definitions. Thus, our realization of this technique aims to automatically generate discriminants for an arbitrary pair of structures and provide them to MULTI.

[2] In the remainder of the paper we often use a pair (Set, Op) consisting of a set Set and a binary operation Op to describe a structure S.

[3] While step 4 is fairly obvious for a human mathematician, it is crucial for a formal proof.

3 The Systems Involved

The proof technique described at the end of the previous section is realized in ΩMEGA's proof planner MULTI. Given two residue class structures $(RS_n^1, \circ^1)$ and $(RS_m^2, \circ^2)$ its task is to show $(RS_n^1, \circ^1) \not\sim (RS_m^2, \circ^2)$ by coordinating the steps 1 to 4 above. To compute a suitable discriminant P, we employ HR [3], a system for theory formation. To obtain a formal proof that P is a discriminant for two arbitrary structures X and Y (i.e., step 4) we use first-order automated theorem provers (ATPs), which we call via TRAMP [9], an interface and transformation system. The proofs that P is a discriminant for the two residue class structures (i.e., that $P(RS_n^1, \circ^1)$ and $\neg P(RS_m^2, \circ^2)$ holds) are done by MULTI itself possibly with the help of the general purpose computer algebra system MAPLE. In this section, we give a short overview of the different systems involved and explain in more detail how the proof planner integrates and coordinates the computations of these systems to assemble an overall proof.

3.1 The HR System

The HR system performs automated theory formation by inventing concepts, making conjectures, proving theorems, and finding counterexamples [3]. The main functionality used for the application to finding discriminants discussed here is concept formation, which is achieved by using production rules which take one (or two) old concepts as input and output a new concept. In particular, we used the following four production rules:

- `Compose`: this composes functions using conjugation.
- `Match`: this equates variables in predicate definitions.
- `Forall`: this introduces existential quantification.
- `Exists`: this introduces universal quantification.

As an example construction which is further discussed later, consider the concept of there being a single element on the diagonal of the multiplication table of an algebra, as is the case for S^1 in Fig. 1. This concept is constructed by HR using the match, forall and exists production rules, as depicted in Fig. 2. In this scenario, two concepts are supplied by the user, namely the concept of an element of the algebra and the multiplication of two elements to give a third. Using the match production rule with the multiplication concept, it invents the notion of multiplying an element by itself. By using this in the forall production rule, it invents the concept of elements which all other elements multiply by

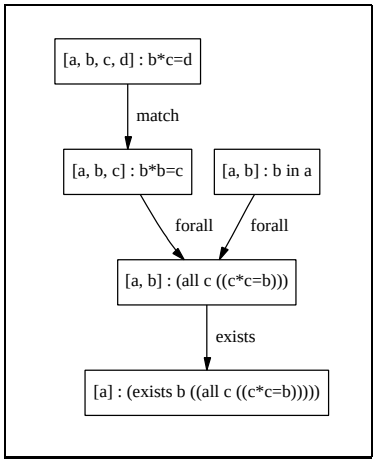

Fig. 2. Example construction

themselves to give. Then, using the exists production rule, HR invents the notion of algebras where there is such an element. As we shall see in section 4, this concept can be used to discriminate between the non-Abelian quasigroups in Fig. 1. For more details of how HR forms concepts, see [6].

There are many parameters that users can experiment with in order to get the best performance out of HR for a particular application. For instance, the user can tell HR to ignore concept formation steps if the arity of the function produced will be greater than a certain limit. HR can tell in advance what arity the function will be for a particular concept formation step involving a particular production rule, and the user can specify a limit for individual production rules.

3.2 Calling Automated Theorem Provers via TRAMP

Calls to ATPs are not directly executed but rerouted via the TRAMP system. TRAMP comprises two functionalities: firstly, it is an interface to several ATPs for first-order logic. For a given problem, it produces input in the formats of the connected systems and runs the systems concurrently. Secondly, TRAMP can transform the output of the ATPs into natural deduction (ND) proofs.

TRAMP currently interfaces a variety of first order ATPs and term rewriting systems. For our problems, TRAMP employs OTTER, BLIKSEM, and SPASS. All three systems are based on the resolution principle and can deal with first-order logic problems with equality. In our scenario, it is necessary that the provers can deal with equality, since the discriminant properties found by HR usually contain equations (see the examples discussed in section 4.1).

3.3 The Multiple-Strategies Proof Planner MULTI

Proof planning [2] considers mathematical theorems as planning problems where an *initial partial plan* is composed of the proof *assumptions* and the theorem as an *open goal*. A proof plan is then constructed with the help of abstract planning steps, called *methods*, that are essentially partial specifications of tactics known from tactical theorem proving. In order to ensure correctness, proof plans have to be executed to generate a sound calculus level proof.

In the ΩMEGA system, the traditional proof planning approach is enriched by incorporating mathematical knowledge into the planning process (see [14] for details). That is, methods can encode general proving steps as well as knowledge particular to a mathematical domain. Moreover, *control rules* provide the possibility of introducing mathematical knowledge on how to proceed in the proof planning process by specifying how to traverse the search space. Depending on the mathematical domain or proof situation, they can influence the planner's behaviour at choice points (e.g., which goal to tackle next or which method to apply next).

ΩMEGA's proof planner MULTI additionally provides a strategic level that extends proof planning. For instance, different planning strategies can implement different proof techniques by specifying particular sets of methods and control

rules. Thus they enable the same problem to be tackled in different ways. Moreover, different strategies for backtracking and other facilities for refining or modifying proof plans can be specified. When more than one strategy is applicable to a problem, MULTI can reason about which strategy to employ and switch strategies during a proof attempt. In particular, the planner can backtrack from applied strategies and thus perform search on the level of strategies.

3.4 Incorporating HR and TRAMP

One way to incorporate specific knowledge into the planning process is by exploiting knowledge implicitly given in specialised external systems. MULTI supports the access of external systems in methods, control rules, and strategies. In general, computations from external systems can be treated in two ways: as hints or as proof steps. The difference is that the soundness of hints is checked by the subsequent proof planning process, which either fails or succeeds for the given hint. On the other hand, to guarantee the soundness of proof steps, special procedures have to be provided which transform the output of external systems into subproofs that ΩMEGA can check.

In our scenario, we have both: the concepts provided by HR are hints and the output of ATPs are proof steps where we employ TRAMP as a corresponding transformation module. MULTI employs HR in a control rule[4]. When HR succeeds in providing a discriminant P for the problem at hand, the control rule triggers the introduction of this P into the proof plan. Note that MULTI can backtrack on different instantiations of P.

The interface to TRAMP is realized in the method *CallTrampM*. When MULTI applies this method to an open goal, a problem consisting of the formula of the goal and the formulas of the proof assumptions is passed to TRAMP. The method is applicable only if one of the provers interfaced by TRAMP succeeds to prove the problem. Then the goal is simply closed by *CallTrampM* without producing further subgoals. Thus at the method level, the proof of this goal is just one step. However, when the correctness of this method application is checked, its expansion results in the ND proof that TRAMP provides as output.

4 Constructing the Proofs

In this section, we describe the whole proof process and the contributions of the individual systems involved in more detail. We illustrate the procedure with the proof that $(\mathbb{Z}_5, \bar{-}) \not\sim (\mathbb{Z}_5, \lambda xy.\bar{2}_5 \bar{*} x \bar{+} y)$ as an example.

The proof procedure is realized with a MULTI proof planning strategy. Among other planning methods, the strategy also contains methods to invoke TRAMP on an appropriately prepared subproblem and a control rule to inject results from

[4] Note that the interface between MULTI and HR is currently not automated. That is, the control rule currently allows the user to supply HR's results. However, we intend to make the interface completely automatic.

HR. When the strategy is applied to a goal of the form $(Set^1, Op^1) \not\sim (Set^2, Op^2)$ MULTI splits this goal into three subgoals of the form:

1. $\mathcal{P}(Set^1, Op^1)$
2. $\neg\mathcal{P}(Set^2, Op^2)$
3. $\forall Set^1, Op^1, Set^2, Op^2.\mathcal{P}(Set^1, Op^1) \wedge \neg\mathcal{P}(Set^2, Op^2)$
$\Rightarrow [(Set^1, Op^1) \not\sim (Set^2, Op^2)]$

Here $\mathcal{P}$ is a newly introduced meta-variable (i.e., a placeholder that has to be replaced with some concrete term later in the proof) for the discriminant. Next, MULTI receives a suitable discriminant P from HR via the afore-mentioned control rule in order to replace the meta-variable $\mathcal{P}$.

For our example, the discriminant[5] property that HR provides is:
$\lambda Set.\lambda Op.\exists x{:}Set.\forall y{:}Set.\ x = Op(y, y)$.
The concrete instantiated subgoals are therefore of the form:

1. $\exists x{:}\mathbb{Z}_5.\forall y{:}\mathbb{Z}_5.x = y \bar{-} y$
2. $\neg\exists x{:}\mathbb{Z}_5.\forall y{:}\mathbb{Z}_5.x = \bar{2}_5 \bar{*} y \bar{+} y$
3. $\forall Set^1, Op^1, Set^2, Op^2.$
$[\exists x{:}Set^1.\forall y{:}Set^1.x = Op^1(y, y)] \wedge \neg[\exists x{:}Set^2.\forall y{:}Set^2.x = Op^2(y, y)]$
$\Rightarrow [(Set^1, Op^1) \not\sim (Set^2, Op^2)]$

Subgoals (1) and (2) are subsequently proved by other MULTI strategies possibly using the computations of the computer algebra system MAPLE. Subgoal (3) is solved with an ATP via TRAMP. In the remainder of this section, we describe the interesting parts of these subproofs in more detail. However, we first explain how we obtain discriminants from HR that are actually useful in our context.

4.1 Obtaining Useful Discriminants from HR

In our scenario, HR needs only to be supplied with the multiplication tables of the example algebras for which it is to find a discriminant, and to be invoked with a flag stating that they are indeed algebraic multiplication tables. Not all possible discriminants are appropriate for our problem. MULTI needs discriminants that are concerned with relations between elements of the sets involved, in particular, existentially or universally quantified statements. For instance, discriminants that involve natural numbers, (e.g., if two structures contain a different number of elements satisfying a certain property) are generally too difficult to handle both for the proof planner and the ATPs. Thus, some of HR's production rules are not appropriate for our application, so they are turned off. In particular, the size production rule that calculates cardinalities of sets – which has been used to find group theory invariants in a different context (see [7] and chapter 12 of [3]) – is not used. We choose only the compose, exists, forall, and match production rules discussed in section 3.1, since they guarantee that only discriminants which are usable in the rest of the proof procedure are found.

[5] A sorted quantification with a variable $x{:}Set$ states that the variable x ranges over the set Set, only.

HR is run for a given number of concept formation steps and told to output those concepts which both specialise the algebras (e.g., Abelian) and discriminate between the two multiplication tables supplied. The output is then transformed into suitable concepts for MULTI. Currently, the interface between MULTI and HR is not automated. Thus, the discriminants are manually ordered and provided to the control rule that establishes the interface between MULTI and TRAMP. We are currently implementing an automatic interface. The discriminants will then be ordered with respect to their complexity, so that MULTI chooses the least complex first, thereby increasing the chances of success during the subsequent proof planning process and the application of the ATPs.

For our example from Fig. 1, some of the discriminants HR discovered are:

- $\exists x. \forall y. x = y \circ y$ [There is a single element on the diagonal of the multiplication table (true for S_1, not true for S_2 and S_3).]
- $\forall x, y. (x \circ x = y \Rightarrow y \circ y = x)$ [There is a symmetry on the diagonal, (true for S_1 and S_3, not true for S_2).]
- $\forall x, y. (x \circ y = x \Rightarrow y \circ x = y)$ [If y is a right identity for x, then x is a right identity for y (true for S_1, not true for S_2 and S_3).]

Here $\circ$ stands for the binary operation of the respective structures S_i. When HR passes the properties to MULTI, they are transformed into a lambda abstraction that can be directly inserted for the meta-variable. For the first of the above discriminants, this results in the term $\lambda Set. \lambda Op. \exists x{:}Set. \forall y{:}Set. x = Op(y, y)$, where the quantified variables are restricted to range over the given set Set, only.

4.2 Subproofs with ATPs

The goal we want to prove with an ATP is generally of the form

$$\forall Set^1, Op^1, Set^2, Op^2. P(Set^1, Op^1) \wedge \neg P(Set^2, Op^2) \Rightarrow [(Set^1, Op^1) \not\sim (Set^2, Op^2)]$$

and contains the defined concept of isomorphism. Providing the usual definition of isomorphism to an ATP was often unsuccessful; that is, the ATPs did not find a proof. Instead the following alternative formalisation is better suited: two structures X and Y are isomorphic iff there are two homomorphisms $h{:}X \to Y$ and $j{:}Y \to X$, such that for all x, $h(j(x)) = x$ and $j(h(x)) = x$.

In order to send the problem with the above definition of isomorphism to the ATPs, the original subproblem has to be adequately rewritten. Therefore, MULTI applies an appropriate theorem from ΩMEGA's knowledge base which results in the following new subgoal to be proved:

$$\forall Set^1, Op^1, Set^2, Op^2. P(Set^1, Op^1) \wedge \neg P(Set^2, Op^2) \Rightarrow \neg(\exists h. \exists j. hom(h{:}(Set^1, Op^1) \to (Set^2, Op^2)) \wedge hom(j{:}(Set^2, Op^2) \to (Set^1, Op^1)) \wedge (\forall x : Set^1. j(h(x)) = x \wedge \forall y : Set^2. h(j(y)) = y))$$

After the expansion of the occurrences of the concept *hom* with the usual definition of homomorphism, the goal can be sent to an ATP.

When trying to close the above subgoal MULTI applies the method *Call-TrampM*. It is applicable when one of the ATPs interfaced by TRAMP succeeds

in finding a proof. TRAMP then returns the corresponding ND proof. The formula passed to TRAMP seems to be a higher-order theorem since it contains quantifications on sets, operations, and the functions h and j. However, when TRAMP calls the connected ATPs it first creates a clause normal form of the given theorem and thereby all the higher-order variables become constants (the theorem is negated for clause normalisation). TRAMP does not use any particular settings for the ATPs but calls them all in their default mode.

On the planning level, the subgoal is simply closed by the application of *CallTrampM*. When the plan is executed, the ND proof provided by TRAMP is the expansion of the *CallTrampM* step. Although TRAMP has the ability to translate refutation proofs into direct ND proofs for our domain the proofs are nevertheless indirect since the indirect argument is an inherent part of the problem. Moreover, in our domain, TRAMP's proofs suffer from the fact that they are low level and can become quite lengthy. For our example, TRAMP produces ND proofs containing between 71 (ND proof transformed from SPASS proof) and 104 steps (from BLIKSEM proof). Thus, the expanded subproofs are usually relatively difficult to understand for the user. Nevertheless, TRAMP provides a proof format that ΩMEGA can directly check.

4.3 Subproofs with MULTI

In the remainder of the proof, we have to prove the two goals $P(Set^1, Op^1)$ and $\neg P(Set^2, Op^2)$, where P is the instantiated discriminant. This part is proved by MULTI using one of two proof planning strategies that have already been implemented to prove simple properties of residue class structures such as associativity, existence of unit elements etc. (see [12] for details).

The first strategy implements an exhaustive case analysis. This is possible since in our problems all quantifiers range over finite sets. The strategy proceeds with the case analysis in two different ways, depending on whether (1) a universal or (2) an existential statement has to be proved. In case (1) a split over all the elements in the set involved is performed and the statement in question is proved for every single element separately. In case (2) the single elements of the set involved are examined stepwise until one is found for which the statement holds.

The idea of the second strategy is to use equational reasoning as much as possible to prove properties of residue classes. Instead of checking the validity of the statements for all possible cases, it tries to reduce goals to equations. Then these equations are passed to the computer algebra system MAPLE to check whether the equality actually holds. If an equation contains meta-variables, these are considered as the variables the equation is to be solved for, and they are supplied to MAPLE as additional arguments.

MULTI always tries to apply the equational reasoning strategy first since it is generally faster and produces shorter proofs. If it fails (e.g., if a goal cannot be reduced to equations), an exhaustive case analysis is used, which is applicable to all occurring problems.

In our example, the goals to prove are:

$$\exists x{:}\mathbb{Z}_5 . \forall y{:}\mathbb{Z}_5 . x = y \bar{-} y \text{ and } \neg\exists x{:}\mathbb{Z}_5 . \forall y{:}\mathbb{Z}_5 . x = \bar{2}_5 \bar{*} y \bar{+} y$$

Algebra	Size	Pairs	Successes	Proportion
Abelian Quasigroups	5	3	3	100.0%
Non-Abelian Quasigroups	5	91	90	98.9%
Non-Abelian Semigroups	5	3	3	100.0%
Abelian Magmas	5	14	13	92.8%
Non-Abelian Magmas	5	630	609	96.7%
Non-Abelian Quasigroups	6	1	1	100.0%
Abelian Semigroups	6	6	6	100.0%
Non-Abelian Semigroups	6	28	25	89.3%
Non-Abelian Quasigroups	10	21	21	100.0%
Abelian Semigroups	10	1	1	100.0%
Non-Abelian Semigroups	10	1	1	100.0%
Abelian Magmas	10	15	15	100.0%
Non-Abelian Magmas	10	3	3	100.0%
Total		817	791	96.8%

Table 1. Proportion of pairs of algebras for which HR found discriminants

The latter goal can only be proved with exhaustive case analysis. MULTI first rewrites the goal by pushing the negation inside: $\forall x{:}\mathbb{Z}_5.\exists y{:}\mathbb{Z}_5.x \neq \bar{2}_5 \bar{*} y \bar{+} y$. Then a case split on the universally quantified variable x over its range $\{\bar{0}_5, \bar{1}_5, \bar{2}_5, \bar{3}_5, \bar{4}_5\}$ is performed. This results in one goal for each of the five elements of the domain. For instance, the resulting goal for $\bar{0}_5$ is $\exists y{:}\mathbb{Z}_5.\bar{0}_5 \neq \bar{2}_5 \bar{*} y \bar{+} y$. To prove this goal, MULTI tries to find a suitable instantiation for y from the set $\{\bar{0}_5, \bar{1}_5, \bar{2}_5, \bar{3}_5, \bar{4}_5\}$ by inserting each possible instantiation for y in succession and attempting to prove the resulting inequality. In our example, it succeeds for $\bar{1}_5$.

The second subgoal, $\exists x{:}\mathbb{Z}_5.\forall y{:}\mathbb{Z}_5.x = y \bar{-} y$, can be proved by the equational reasoning strategy. Thereby, MULTI first decomposes the quantifiers which results in the goal $cl_5(z) = cl_5(c) \bar{-} cl_5(c)$ where z is a new meta-variable and c is a new constant. Since this goal is a statement on congruence classes, it is transformed into the corresponding goal on integers: $z \bmod 5 = (c - c) \bmod 5$, and passed to MAPLE. MAPLE applies its function `msolve` to compute that the equation holds in general for $z = 0$. This result is used as instantiation for z in the proof plan and the goal is closed.

5 Results

To test HR's effectiveness at finding discriminants for arbitrary pairs of algebraic objects, we used 817 pairs over 6 different algebras (of sizes 5, 6, and 10). For each pair, HR was allowed 500 concept formation steps in order to find discriminants. We allowed HR to use the four production rules previously mentioned, and after some experimentation, we chose the following function arity limits: compose(3), exists(1), forall(2) and match(3). The average session took around 22 seconds on a 500Mhz pentium machine. The results are given in table 1. We note that HR found discriminants for nearly 97% of the pairs given to it, which was a higher success rate than we anticipated given the short session lengths we allowed for these tests. On average, HR found 20 discriminants for each pair, and in total, 517 distinct concepts were used as discriminants.

$M^1 = (\mathbb{Z}_5, \lambda xy.(x \bar{-} \bar{1}_5) \bar{*} (y \bar{*} \bar{2}_5))$

M^1	$\bar{0}_5$	$\bar{1}_5$	$\bar{2}_5$	$\bar{3}_5$	$\bar{4}_5$
$\bar{0}_5$	$\bar{0}_5$	$\bar{3}_5$	$\bar{1}_5$	$\bar{4}_5$	$\bar{2}_5$
$\bar{1}_5$	$\bar{0}_5$	$\bar{0}_5$	$\bar{0}_5$	$\bar{0}_5$	$\bar{0}_5$
$\bar{2}_5$	$\bar{0}_5$	$\bar{2}_5$	$\bar{4}_5$	$\bar{1}_5$	$\bar{3}_5$
$\bar{3}_5$	$\bar{0}_5$	$\bar{4}_5$	$\bar{3}_5$	$\bar{2}_5$	$\bar{1}_5$
$\bar{4}_5$	$\bar{0}_5$	$\bar{1}_5$	$\bar{2}_5$	$\bar{3}_5$	$\bar{4}_5$

$M^2 = (\mathbb{Z}_5, \lambda xy.(x \bar{+} \bar{1}_5) \bar{*} (y \bar{*} \bar{2}_5))$

M^2	$\bar{0}_5$	$\bar{1}_5$	$\bar{2}_5$	$\bar{3}_5$	$\bar{4}_5$
$\bar{0}_5$	$\bar{0}_5$	$\bar{2}_5$	$\bar{4}_5$	$\bar{1}_5$	$\bar{3}_5$
$\bar{1}_5$	$\bar{0}_5$	$\bar{4}_5$	$\bar{3}_5$	$\bar{2}_5$	$\bar{1}_5$
$\bar{2}_5$	$\bar{0}_5$	$\bar{1}_5$	$\bar{2}_5$	$\bar{3}_5$	$\bar{4}_5$
$\bar{3}_5$	$\bar{0}_5$	$\bar{3}_5$	$\bar{1}_5$	$\bar{4}_5$	$\bar{2}_5$
$\bar{4}_5$	$\bar{0}_5$	$\bar{0}_5$	$\bar{0}_5$	$\bar{0}_5$	$\bar{0}_5$

Fig. 3. The multiplication tables of two non-Abelian magmas.

In the sets of discriminants HR constructed, there were usually some of a complexity similar to those discriminants discussed in section 4.1. In fact, one of the most complex discriminants HR found was for the two non-Abelian magmas M^1 and M^2 given in Fig. 3. HR constructed only two discriminants for M^1 and M^2, including: $\exists x_{\blacksquare}(x \circ x = x \wedge \forall y_{\blacksquare}(y \circ y = x \Rightarrow y \circ y = y))$. This states that there exists an idempotent element, x, such that any other element which squares to give x is itself idempotent. After some thought, it is obvious that this means that there must be an idempotent element which appears only once on the diagonal. Such an element exists in M^2 (the element is $\bar{2}_5$), but not in M^1.

The relative simplicity of the discriminants facilitates the subsequent proofs of MULTI and the ATPs. In particular, for the latter, it is usually hard to predict whether or not they will succeed for a given discriminant. Here, a large set of different discriminants can increase the chance of success, since MULTI can backtrack if the ATPs cannot come up with a proof for one discriminant and instantiate another one. So far, we have tested the planning and ATP side with roughly 100 examples for which HR found discriminants. All tested examples were successfully proved. Among the ATPs, SPASS and BLIKSEM performed best on our problems. In particular, we also proved the example of the two non-Abelian magmas with the discriminant given above. The resulting proof plan consists of 154 steps. The ATP part was proved by BLIKSEM, which was transformed by TRAMP into an ND proof containing 112 steps.

A comparison of attempts to solve problems with structures using $\mathbb{Z}_5$ or $\mathbb{Z}_6$ as opposed to structures using $\mathbb{Z}_{10}$ shows that the incorporated systems depend on the cardinality of the involved residue class structures to different degrees. In our experiments, HR's performance did not vary significantly when applied to problems with $\mathbb{Z}_5$ or $\mathbb{Z}_{10}$. That is, HR solved 100% of the problems of size 10 (see Table 1), but more importantly, it took roughly the same time to provide the solutions. This is because, in this application, HR works with only two structures at a time, which is a much smaller amount of data than it usually works with, so the difference between size 5 and 10 algebras was minimal.

The ATPs were also not affected by the cardinality of the involved structures simply because the subproblem they are responsible for does not contain the residue class structures. The complexity of the problem part for the ATPs depends only on the complexity of the discriminants provided by HR. However,

in our experiments, we found that the complexity of HR's discriminants does not depend on the cardinality of the involved residue class structures.

Unlike HR and the ATPs, MULTI's performance depends on both the cardinality of the residue class sets, and the strategy applied. When performing equational reasoning, MULTI solves a subproblem with set $\mathbb{Z}_5$ in the same number of steps as the same subproblem with set $\mathbb{Z}_{10}$. On the other hand, MULTI's performance considerably differs when it applies the exhaustive case analysis strategy. A subproblem with l nested quantifications for $\mathbb{Z}_5$ results in 5^l cases, whereas for $\mathbb{Z}_{10}$, it results in 10^l cases. Although MULTI prefers the application of the equational reasoning, it is never the case that both of MULTI's subproblems, $P(Set^1, Op^1)$ and $\neg P(Set^2, Op^2)$, are solvable by this strategy.

6 A Comparison with a Different Approach

We have also experimented with the model generator SEM [17] to construct discriminants. Here, properties are created in ΩMEGA and SEM tests whether each property is a discriminant. That is, SEM finds models for each of the two involved structures with respect to the created property. A property is a candidate for a discriminant if SEM succeeds for one structure and fails for the other one. The possible properties are constructed as follows. The basic construction element is the equation $x = Op(y, z)$. Associated with this equation are all formulas/properties that result (1) from quantifying the x, y, z either existentially or universally (e.g., $\forall x. \exists y. \exists z. x = Op(y, z)$ or $\exists x. \forall y. \forall z. x = Op(y, z)$ etc.). Further properties are generated by (2) permuting the sequence of the quantifications for x, y, z (e.g., $\forall y. \exists z. \exists x. x = Op(y, z)$ etc.) and by (3) making some variables equal (e.g., $\exists y. \forall x. x = Op(y, x)$). More complicated properties result from the combination of several copies of the basic construction element by $\wedge, \vee, \Rightarrow$, for instance, $x_1 = Op(y_1, z_1) \Rightarrow x_2 = Op(y_2, z_2)$ or $x_1 = Op(y_1, z_1) \wedge (x_2 = Op(y_2, z_2) \Rightarrow x_3 = Op(y_3, z_3))$ etc. The properties associated with such combined equations are again constructed using the mechanisms (1), (2), and (3) etc.

For a residue class structure $(RS_n, \circ)$ and a property p, SEM is passed the multiplication table of the operation of the residue class structure together with the clauses resulting from the normalization of the property p. The multiplication table for RS_n is encoded as a set of n^2 equations of the form $c = Op(a, b)$ where c is the result of $a \circ b$. Then SEM is asked to find a model for this input. We tried to find discriminants for two given residue class structures by either systematically or randomly checking the properties in the space of our construction mechanism. However, it turned out that we could only find very simple discriminants, such as $\lambda Set. \lambda Op. \exists x{:}Set. \forall y{:}Set. x = Op(y, y)$. More complicated discriminants like the discriminant $\exists x. (x * x = x \wedge \forall y. (y \circ y = x \Rightarrow y \circ y = y))$ discussed in section 5 are in our search space but are out of the reach of this approach[6].

[6] Note that even for the combination of two copies of the basic construction element (e.g., $x_1 = y_1$ *op* $z_1 \Rightarrow x_2 = y_2$ *op* z_2) the number of possible properties resulting from the construction mechanisms (1), (2), and (3) is roughly 100000.

The problems of the approach using SEM are twofold. First, we search in an enormous space of possible candidates and second we search blindly since we were not able to guide the search by powerful heuristics. Restricted to the 4 production rules mentioned, HR searches a similar space, but its search mechanism is a little more sophisticated, as it recognises two things. Firstly, if it constructs a concept with no examples, then no further concepts are built from that, as they will also have no examples (and hence will be useless as a discriminant). Secondly, if it constructs a concept with the same examples as a previous concept, the new concept is discarded, to reduce redundancy. Such simple empirical checks are very powerful in reducing the amount of search HR undertakes.

HR's success was achieved using only one of many mechanisms available to it: a limited search, where certain search steps are taken off the agenda if they would result in functions of too high an arity. If the problems had been more difficult, there are two additional mechanisms we could have experimented with, namely: (i) a heuristic search, which measures the concepts in various ways and builds new concepts from the most interesting old ones first [5] and (ii) a forward look ahead mechanism, which can tell in advance whether the application of up to three concept formation steps will lead to a concept which achieves a particular categorisation task, in our case finding an invariant, as discussed at length in [6].

7 Conclusion

To produce more sophisticated systems to tackle increasingly difficult problems, it is necessary to combine programs so that the whole is greater than the sum of the parts. We have presented an example here of a fruitful cooperation of heterogeneous mathematical systems to prove non-isomorphism theorems in the residue class domain. The cooperation is essentially orchestrated by a proof planner, MULTI, which uses a theory formation system, HR, to construct appropriate discriminants for given structures. The proof is then planned with the help of automated theorem provers and sometimes a computer algebra system.

The problem of identifying a discriminant for two objects is a machine learning problem, which could, in theory, be solved by a program such as Progol [15], which uses Inductive Logic Programming to identify a concept which correctly categorises a set of positive and negative examples. However, as mentioned in [4], this may be difficult in practice because we only supply a single positive and a single negative example, which would suggest that the amount of compression in a concept would not be high enough to be suggested as a viable solution.

Currently, we call HR manually and supply the proof planner with the appropriate information, but we are working on automating this communication. Nevertheless, we have already tested the cooperation with a number of examples and HR proved successful for the vast majority of these examples. Moreover, HR's discriminants could be handled both by the proof planner and the automated theorem provers successfully even for relatively complicated discriminants.

References

1. C. Benzmüller, L. Cheikhrouhou, D. Fehrer, A. Fiedler, X. Huang, M. Kerber, M. Kohlhase, K. Konrad, E. Melis, A. Meier, W. Schaarschmidt, J. Siekmann, and V. Sorge. ΩMega: Towards a Mathematical Assistant. In *Proceedings of the 14th International Conference on Automated Deduction (CADE–14)*, volume 1249 of *LNAI*, pages 252–255. Springer Verlag, Germany, 1997.
2. A. Bundy. The Use of Explicit Plans to Guide Inductive Proofs. In *Proceedings of the 9th International Conference on Automated Deduction (CADE–9)*, volume 310 of *LNCS*, pages 111–120. Springer Verlag, Germany, 1988.
3. S. Colton. *Automated Theory Formation in Pure Mathematics.* PhD thesis, Department of Artificial Intelligence, University of Edinburgh, 2000.
4. S Colton. An application-based comparison of automated theory formation and inductive logic programming. *Linkoping Electronic Articles in Computer and Information Science (special issue: Proceedings of Machine Intelligence 17)*, forthcoming, 2002.
5. S Colton, A Bundy, and T Walsh. On the notion of interestingness in automated mathematical discovery. *International Journal of Human Computer Studies*, 53(3):351–375, 2000.
6. S. Colton, A. Bundy, and T. Walsh. Automatic identification of mathematical concepts. In *Proceedings of the 17th International Conference on Machine Learning (ICML2000)*, pages 183–190. Morgan Kaufmann, USA, 2001.
7. S Colton, S Cresswell, and A Bundy. The use of classification in automated mathematical concept formation. In *Proceedings of SimCat 1997: An Interdisciplinary Workshop on Similarity and Categorisation.* University of Edinburgh, 1997.
8. The GAP Group, Aachen, St Andrews. *GAP – Groups, Algorithms, and Programming, Version 4*, 1998. http://www-gap.dcs.st-and.ac.uk/~gap.
9. A. Meier. Tramp: Transformation of Machine-Found Proofs into ND-Proofs at the Assertion Level. In *Proceedings of the 17th International Conference on Automated Deduction (CADE–17)*, volume 1831 of *LNAI*, pages 460–464. Springer Verlag, Germany, 2000.
10. A. Meier, M. Pollet, and V. Sorge. Classifying Isomorphic Residue Classes. In *Proceedings of the 8th International Workshop on Computer Aided Systems Theory (EuroCAST 2001)*, volume 2178 of *LNCS*, pages 494–508. Springer Verlag, Germany, 2001.
11. A. Meier, M. Pollet, and V. Sorge. Comparing Approaches to Explore the Domain of Residue Classes. *Journal of Symbolic Computations*, 2002. forthcoming.
12. A. Meier and V. Sorge. Exploring Properties of Residue Classes. In *Proceedings of the CALCULEMUS-2000 Symposium*, pages 175–190. AK Peters, USA, 2001.
13. E. Melis and A. Meier. Proof planning with multiple strategies. In *Proceedings of the First International Conference on Computational Logic*, volume 1861 of *LNAI*. Springer Verlag, Germany, 2000.
14. E. Melis and J. Siekmann. Knowledge-Based Proof Planning. *Artificial Intelligence*, 115(1):65–105, 1999.
15. S Muggleton. Inverse entailment and Progol. *New Generation Computing*, 13:245–286, 1995.
16. D. Redfern. *The Maple Handbook: Maple V Release 5.* Springer Verlag, , Germany, 1999.
17. J. Zhang and H. Zhang. SEM: a System for Enumerating Models. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 298–303. Morgan Kaufmann, USA, 1995.