Available online at www.sciencedirect.com
SCIENCE DIRECT®

Electronic Notes in Theoretical Computer Science

ELSEVIER Electronic Notes in Theoretical Computer Science 125 (2005) 87–101

www.elsevier.com/locate/entcs

The TM System for Repairing Non-Theorems

Simon Colton

Department of Computing
Imperial College, London
United Kingdom
sgc@doc.ic.ac.uk

Alison Pease

School of Informatics
University of Edinburgh
United Kingdom
alisonp@dai.ed.ac.uk

Abstract

We describe a flexible approach to automated reasoning, where non-theorems can be automatically altered to produce proved results which are related to the original. This is achieved in the TM system through an interaction of the HR machine learning program, the Otter theorem prover and the Mace model generator. Given a non-theorem, Mace is used to generate examples which support the non-theorem, and examples which falsify it. HR then invents concepts which categorise these examples and TM uses these concepts to modify the original non-theorem into specialised theorems which Otter can prove. The methods employed by TM are inspired by the piecemeal exclusion, strategic withdrawal and counterexample barring methods described in Lakatos's philosophy of mathematics. In addition, TM can also determine which modified theorems are likely to be interesting and which are not. We demonstrate the effectiveness of this approach by modifying non-theorems taken from the TPTP library of first order theorems. We show that, for 98 non-theorems, TM produced meaningful modifications for 81 of them. This work forms part of two larger projects. Firstly, we are working towards a full implementation both of the reasoning and the social interaction notions described by Lakatos. Secondly, we are aiming to show that the combination of reasoning systems such as those used in TM will lead to a new generation of more powerful AI systems.

Keywords: Automated theorem modification, automated reasoning, model generation, machine learning, automated theory formation, philosophy of mathematics.

doi:10.1016/j.entcs.2004.07.021

1 Introduction

Mathematics has developed in a much more organic way than its rigid textbook presentation of definition-theorem-proof would suggest. Automated theorem proving systems more closely reflect the textbook notion of mathematics than a developmental approach. In particular, most deduction systems are designed either to prove results if they are true, or find counterexamples if they are false, but not both. System designers also assume that the concepts mentioned in the conjecture are correctly defined and actually relate to the mathematical notions the user is interested in. Clearly, the adoption of these assumptions is not conducive to the kind of exploration more common in research mathematics, in which concept definitions change and become more sophisticated, and flawed conjectures and proofs are gradually refined. Hence, it is time to increase the flexibility of reasoning systems to better handle ill-specified problems.

We describe here the development of the Theorem Modifier (TM) system. This takes a set of axioms and a conjecture in first order logic and tries to prove it. If this fails, TM attempts to modify the conjecture into a set of theorems which it can prove. To achieve this flexibility, TM combines the power of three automated reasoning systems, namely the HR machine learning program [1], the Otter theorem prover [11] and the Mace model generator [12]. As described in §3, TM uses these systems in ways prescribed in the philosophy of mathematics developed by Lakatos [9]. In particular, TM performs counterexample-barring, piecemeal exclusion and strategic withdrawal. These techniques are further explained in §2. As a simple example of TM working, given the non-theorem that all groups are Abelian, it states that it cannot prove the original result, but it has discovered that self-inverse groups are Abelian. To evaluate this approach, in §4, we describe how TM successfully found meaningful modifications to 81 of 98 non-theorems derived from the TPTP library of first order theorems [16].

The development of the TM system forms part of two larger projects. Firstly, we are working towards a full implementation both of the reasoning and the social interaction notions described by Lakatos in [9]. Secondly, we are aiming to show that the combination of reasoning systems such as those used in TM will lead to a new generation of AI systems which are able to solve problems which individual techniques cannot.

2 Background

The way in which TM forms modified theorems is inspired by the notions expressed in the philosophy of mathematics presented by Imre Lakatos [9], as described in §2.1 below. The implementation of these ideas is heavily dependent on third party software, in particular the Otter, Mace and HR programs. Of these, HR is the least well known, so we describe this in §2.2.

2.1 Lakatos's Philosophy of Mathematics

Our inspiration for TM comes from Lakatos, who presented a fallibilist approach to mathematics, in which proofs, conjectures and concepts are fluid and open to negotiation [9]. Lakatos strongly criticised the deductivist approach in mathematics, which presents definitions, axioms and theorem statements as immutable ideas which come from nowhere into a mathematician's empty mind. Rather than a mysterious and ever-increasing set of truths, Lakatos saw mathematics as an adventure in which – via patterns of analysis which he categorised into various methods – conjectures and proofs are gradually refined but never certain. He rejected the view that discovery in mathematics is essentially irrational and should be left to the psychological arena, as championed by, for instance, Popper [15]. Instead, he outlined a heuristic approach which holds that mathematics progresses by a series of primitive conjectures, proofs, counterexamples, proof-generated concepts, modified conjectures and modified proofs. Lakatos demonstrated his argument using case studies including the development of Euler's conjecture that for any polyhedron, the number of vertices (V) minus the number of edges (E) plus the number of faces (F) equals two.

Lakatos's treatment of exceptions is noteworthy for two reasons. Firstly, he highlights their existence in mathematics – traditionally thought of as an exact subject. Secondly, he shows how exceptions, rather than simply being annoying problem cases which would force a mathematician to abandon a conjecture, can be used to further knowledge. He does this via two methods; *piecemeal exclusion* and *strategic withdrawal*. Piecemeal exclusion works by generalising from a counterexample to a class of counterexamples and then excluding this class from the faulty conjecture. For instance, Lakatos showed how, by examining the hollow cube which is a counterexample to Euler's conjecture, mathematicians modified the conjecture to 'for any polyhedron without cavities, $V - E + F = 2$' [9]. Put formally, suppose that we have the conjecture $\forall\ x\ (A(x) \Rightarrow B(x))$, a set of counterexamples N such that $\forall\ x \in N, A(x) \wedge \neg B(x)$, and a set of positive examples P such that $\forall\ x \in P, A(x) \wedge B(x)$. To perform piecemeal exclusion, find a concept $\mathcal{C}$ such

that $\forall\ x \in N,\ C(x)$, and $\forall\ x \in P,\ \neg C(x)$, then modify the conjecture to: $\forall\ x\ (\neg C(x) \wedge A(x)) \Rightarrow B(x)$. When there is only one counterexample and no simply expressed concept which covers it, piecemeal exclusion extends to *counterexample-barring*, in which the counterexample is explicitly forbidden in a modified conjecture, i.e., given a single counterexample $x_1 \in N$, one modifies the conjecture to: $\forall\ x \neq x_1\ (A(x) \Rightarrow B(x))$.

Strategic withdrawal works by considering the examples supporting a conjecture, finding a concept which covers a subset of these, and limiting the domain of the conjecture to that of the concept. For instance, by examining the supporting examples of Euler's conjecture, such as the cube, tetrahedron and octahedron, mathematicians retreated to the 'safe' domain of convex polyhedra (*i.e.* polyhedra whose surface is topologically equivalent to the surface of a sphere). Put formally, given the above conjecture, set of supporting examples P and counterexamples N, first find a concept $\mathcal{C}$ such that $\forall\ x \in P,\ C(x)$, and $\forall\ x \in N,\ \neg C(x)$, then modify the conjecture to: $\forall\ x\ (C(x) \wedge A(x)) \Rightarrow B(x)$.

Clearly, an implementation of a theorem modification system along the lines suggested by Lakatos requires three core functionalities. Firstly, an ability to prove theorems is required. We achieved this by incorporating the Otter program [11] into the system. Otter is a powerful first order resolution theorem prover which has been used for many discovery tasks in algebraic domains, e.g., [13]. Secondly, an ability to generate counterexamples to non-theorems is required. We achieved this by incorporating the Mace program [12] into the system. Mace is a powerful model generator which employs the Davis-Putnam method for generating models to first order sentences. Thirdly, an ability to suggest modifications to non-theorems in the light of counterexamples is required. We achieved this by incorporating the HR program into the system. HR is described below.

2.2 The HR System

HR is named after the mathematicians Hardy and Ramanujan, and the core functionality of this system is described in [1]. HR performs descriptive induction to form a theory about a set of objects of interest which are described by a set of background concepts, as detailed further in [3]. This is in contrast to predictive learning systems which are used to solve the particular problem of finding a definition for a target concept. The theories HR produces contain concepts which relate the objects of interest; conjectures which relate the concepts; and proofs which explain the conjectures. Theories are constructed via theory formation steps which attempt to construct a new concept. HR builds new concepts from old ones using a set of 15 generic production rules [4] which include:

• The *exists* rule: this adds existential quantification to the new concept's definition

• The *negate* rule: this negates predicates in the new definition

• The *match* rule: this unifies variables in the new definition

• The *compose* rule: this takes two old concepts and combines predicates from their definitions in the new concept's definition

For a more formal description of these production rules, and the others that HR uses, see [3] or [4].

For each concept, HR calculates the set of examples which have the property described by the concept definition. Using these examples, the definition, and information about how the concept was constructed and how it compares to other concepts, HR estimates how interesting the concept is [6], and this drives a heuristic search. As it constructs concepts, it looks for empirical relationships between them, and formulates conjectures whenever such a relationship is found. In particular, HR forms equivalence conjectures whenever it finds two concepts with exactly the same examples, implication conjectures whenever it finds a concept with a proper subset of the examples of another, and non-existence conjectures whenever a new concept has an empty set of examples. HR is also able to make near-conjectures whenever the relationship has only a few counterexamples. To attempt to determine the truth of each conjecture, they are passed to a third party theorem prover and a third party counterexample finder (usually Otter and Mace, but there are interfaces to other reasoning systems [17]). HR also works hard to break the conjectures into lemmas which are easier to prove, and it will also extract prime implicates which may be more interesting to the user [2].

HR has been used for a variety of discovery projects in mathematics. It has been particularly successful in number theory [5] and algebraic domains [14]. Moreover, we have used HR to improve the abilities of Artificial Intelligence systems, most notably constraint solvers [7], and we are currently extending HR to perform discovery tasks in other scientific domains, in particular bioinformatics. While we have used HR to generate first order conjectures [8], the application described in this paper is the first one in which we have applied HR to the problem of *proving*, rather than generating, theorems.

3 Automated Theorem Modification

Users supply TM with a conjecture of the form: $A \Rightarrow C$ where A is a conjoined set of axioms which describe the domain they are working in, and C is the

statement of the conjecture they wish to prove/modify/disprove. The theorem is supplied in Otter first-order syntax, which means that C must be negated, as Otter will derive a contradiction using resolution. TM assumes that C is placed in the last line of input, preceded by a line per axiom. We hope to relax such restrictions in future versions of the program. For the purposes of this paper, we also assume that we are working in an algebraic domain, where algebraic objects comprise a set of elements and a set of operators relating those elements which are constrained as prescribed by the axioms. An example algebra is group theory, where there is a single operator which satisfies the associativity, identity and inverse axioms.

3.1 Forming Modified Theorems

How the TM program operates can be characterised by how and when it calls the Otter, Mace and HR programs, and how it implements the piecemeal exclusion, strategic withdrawal and counterexample-barring methods described in §2. To begin with, TM checks whether the conjecture is true, i.e., $A \Rightarrow C$. It does this by invoking Otter for a period of time specified by the user, and if Otter is successful, this is reported to the user and TM stops. If the theorem cannot be proved by Otter in the time given, TM then uses Otter to attempt to prove that the negation of C follows from A. For reasons we shall see later, if the negation of the theorem is true, then TM will not be able to modify this conjecture using its current techniques.

Next, TM checks whether the conjecture is true if and only if the objects in the domain (which are all algebraic objects) are trivial – in the sense that they have only one element – and whether the conjecture is true if and only if the objects are non-trivial. To do this, Otter is asked to prove: $A \Rightarrow ((\forall\ a, b\ (a = b)) \Leftrightarrow C)$ and $A \Rightarrow ((\exists\ a, b\ (a \neq b)) \Leftrightarrow C)$ respectively. If either can be proved, then TM returns the modified theorem that the conjecture is true for trivial/non-trivial algebras only. These are special cases, and checking for them is in line with Lakatos's counterexample-barring. If TM were to follow Lakatos's advice directly, then it would first find counterexamples to the theorem and try to prove that if they are excluded from the conjecture, it is true. However, in all but a few cases, we have found that Otter is not good at proving such results, as describing the models to be excluded leads to a great number of first order sentences being added to the input file for Otter. The exception, of course, is when the algebra to be excluded is trivial, as we have seen above that this can be simply stated. However, in algebraic domains, theorems which are true for all but the trivial algebra are quite rare. In fact, the opposite is often true: the theorem is true *only for* the trivial algebra. For these reasons, we decided that having TM check initially

for these two simple modifications was a better idea than implementing full counter-example barring techniques.

If none of these preliminary checks have been successful, then the conjecture is either a non-theorem, or is too difficult for Otter to prove in the time available. In either case, this presents an opportunity to modify the theorem in order to enable Otter to prove it. We have so far concentrated on modifying a conjecture by specialising it, i.e., adding in extra conditions which enable Otter to prove the modified theorem. To do this, TM first finds some example algebras which support the conjecture, by using Mace to generate models which satisfy A and for which C holds. Mace is then used to generate some examples which contradict the conjecture, i.e., models which satisfy A but which break the conjecture C. Mace is given a limit for both time and size. Normally, we ask Mace to find an example of size 1, an example of size 2, etc., up to size 8, and that it can spend 10 seconds on each search. For instance, when we give TM the false conjecture that all groups are Abelian, it uses Mace to find an example Abelian group for each size 1 to 8, which support the conjecture. However, it also finds a non-Abelian group of size 6 and a non-Abelian group of size 8, which falsify the conjecture.

The supporting and falsifying examples generated by Mace are given as the objects of interest to a session using HR. HR is also supplied with the file containing the statement of the conjecture in Otter format. From this, it extracts the background concepts in the domain, e.g., in group theory, HR would extract the concept of groups, elements, multiplication, identity and inverse. These form the basis of the theory HR forms, i.e., all concepts it produces will be derived from these. TM then uses HR to form a theory for a user-set number of theory formation steps, usually taken to be between 1000 and 5000. In this time, HR generates many concepts which can be interpreted as specialisations of the algebra, such as Abelian groups, self-inverse groups, etc. For instance, in group theory, given the groups up to size 8 as input, in 5000 steps, HR generates 37 specialisations of the concept of group.

From the theory produced by HR, TM identifies all the specialisation concepts and extracts those which describe only algebras that support the conjecture. For example, in the session associated with the non-theorem that all groups are Abelian, amongst others, HR invents the concept of groups which are self inverse, i.e., $\forall\ a\ (a = a^{-1})$. It turns out that these form a subset of the examples which supported the conjecture, and hence TM extracts this from HR's theory. For each extracted specialisation, M, TM forms the modified conjecture: $(A \wedge M) \Rightarrow C$ by adding M to the axioms. Otter is invoked to see which of these modifications can be proved, and any which are proved are presented to the user. Note that, in addition to the specialisations that

HR produces, TM also extracts any concepts which have been conjectured to be logically equivalent to a specialisation – these concepts are not normally allowed into the theory as distinct items, but HR records the conjectured equivalence of the definitions. This functionality is turned on by default, but the user can set a flag to stop it happening, which will produce faster results (as fewer calls to Otter will be made), but has the potential to miss interesting modifying specialisations.

3.2 Identifying Uninteresting Modifications

Unfortunately, there are a number of reasons why the modifications generated by this process can be uninteresting for the user. TM takes care to discard any it can prove to be uninteresting, and highlights any which have a greater chance than normal of being uninteresting. In particular, some specialisations that HR produces are true only of the trivial algebra. As most conjectures are also true of the trivial algebra, the modifications usually hold, but are uninteresting. For instance, the modified conjecture: "all groups which are the trivial group are Abelian" holds very little interest. Hence, whenever a modification has been proved, and the examples satisfying the definition of the specialisation M in the modification amount to just the trivial algebra, TM invokes Otter to check whether: $A \Rightarrow (M \Leftrightarrow (\forall\, a, b\, (a = b)))$ It is unlikely, but not impossible that such re-definitions of the trivial algebra will be interesting to the user (for instance, the re-definition might contain an unusual combination of background concepts). Hence, in TM's output, the modification is set-aside from the others, but it is not discarded.

Another problem arises when HR derives concepts which are re-definitions of the conjecture statement. Obviously, adding this to the axioms would make the conjecture trivially true, e.g., all Abelian groups are Abelian. Hence, for every specialisation, M, where *every* supporting example has the property prescribed by M, TM uses Otter to try to prove: (i) $M \Leftrightarrow C$ (ii) $A \Rightarrow (M \Leftrightarrow C)$ and (iii) $M \Rightarrow C$. Often M is just a simple restatement of C, and of no interest, but it sometimes happens that the equivalence of C and M is quite surprising and non-trivial to prove, hence the modification is valid. Hence, if TM proves any of the three results above, it presents the modification to the user separately, and provides the result as a possible indication of why the modification is true and a caution that it may be uninteresting because the specialisation trivially proves the conjecture.

This process of modifying conjectures by specialising them is an implementation of Lakatos's strategic withdrawal method, whereby a concept which excludes all of the counterexamples is discovered and the conjecture is specialised to only apply to examples satisfying that concept. Note also that

when HR uses the negate rule, which TM instructs it to, for every specialisation M, the negation $\neg M$ will also be produced. Hence, if the examples of M contained all the *falsifying* examples for the conjecture, then $\neg M$ would describe a subset of the supporting examples, and hence would be used in a modification attempt. Recalling that the piecemeal exclusion strategy involves finding a concept which covers all the counterexamples (and possibly more), then excluding the concept from the conjecture, we see that TM is also using piecemeal exclusion to form the modifications.

3.3 Summary of Theorem Modification

To summarise, in our running example that all groups are Abelian, TM undertakes the following process. Firstly, it tries and fails to prove that the conjecture is true already, and similarly fails to prove that the negation of the conjecture follows from the axioms (i.e., it fails to prove that all groups are non-Abelian). If the latter were true, then no amount of specialisation would improve matters. TM also fails to prove that a group is Abelian if and only if it is trivial, and that a group is Abelian if and only if it is non-trivial. It then employs Mace to generate some Abelian groups which support the conjecture and some non-Abelian groups which falsify the conjecture. Both sets of examples are given, along with the conjecture statement as input to HR, which forms a theory of groups containing many specialisations of the notion of group. From this theory, TM extracts all those specialisations which describe only groups which support the conjecture. When using one of these, namely self-inverse groups, in a modified conjecture, Otter proves the theorem and TM reports that it can prove that self-inverse groups are Abelian, even though the original conjecture is false. In contrast to the usual proof-or-fail output from a theorem prover, TM outputs 5 different types of result:

- The original conjecture is true: $A \Rightarrow C$
- The negation of the conjecture is true: $A \Rightarrow \neg C$
- The conjecture is true only for trivial algebras $(A \Rightarrow C) \Leftrightarrow Triv$
- It is true only for non-trivial algebras $(A \Rightarrow C) \Leftrightarrow \neg Triv$
- The original conjecture is false, but various modifications of it are true, $(A \wedge M) \Rightarrow C$

In the latter case, when appropriate, TM can also warn the user that the modification may be trivially true because either M is only true of the trivial algebra, or because one of the following lemmas holds: $M \Leftrightarrow C$, $A \Rightarrow (M \Leftrightarrow C)$ or $M \Rightarrow C$.

4 Testing and Evaluation

We used the TPTP library [16] to supply a set of non-theorems for experiments designed to test the hypothesis that TM can find meaningful modifications to non-theorems. We looked at four categories within TPTP, namely GRP (groups), FLD (fields), RNG (rings) and COL (combinatory logic). Unfortunately, we found only 9 non-theorems which were suitable, because (a) there aren't many non-theorems (b) many were actually just statements of axioms for which models can be found and (c) many were not in the form of axioms followed by conjecture, e.g., there are many non-theorems stating that one set of axioms is not equivalent to another set.

In order to provide a more substantial test set, we took theorems from the above TPTP categories and altered them to become non-theorems. The alterations included (i) removing axioms (ii) changing/removing quantifiers (iii) altering variables and constants and (iv) altering bracketing. In this fashion, we produced 158 altered theorems, which we used alongside the 9 proper non-theorems. We found that 30 of our altered TPTP theorems were still theorems (TM told us this). Mace produced the same examples to both support and falsify the conjecture, for 39 of the remaining 137 non-theorems. This was due to constants such as an identity element being used in the conjecture statement without reference in the axioms, or to variables being instantiated differently. We removed these non-theorems from the test set, leaving us with a core of 98 non-theorems.

In addition to testing the effectiveness of TM, we also wanted to determine whether any alterations to the setup would improve the performance. It was clear from an early stage that giving Mace extra time and range did not improve matters, as it only found a few more examples which did not affect the specialising concepts that HR found. Also, we experimented by giving Otter more time, but we have not seen any evidence that this improves performance of TM – it is often the case that if a prover is going to solve a problem, it will do so quickly, and giving a little extra time will not help for more difficult problems. Hence, we concentrated on altering the way in which we ran HR. We ran three sessions using TM to attempt to modify each of the 98 non-theorems. Otter and Mace were given 10 seconds, with Mace looking for examples up to size 8, and HR was allowed 1000 theory formation steps in the first two sessions, and 3000 steps in the third session. In the first session, however, the ability to use equivalence conjectures to harvest specialisations was turned off. The results are presented in table 1.

In the sessions, we found that in many cases, the only modifications came with a caution that the specialisation may trivially make the theorem true. However, in around two thirds of these cases, upon looking at the modification,

Session	1	2	3
Equivalent to trivial algebra	24	24	24
No valid modifications	11	10	9
Only redefinition modifications	8	8	8
Valid modifications with caution	18	18	18
Valid modifications no caution	37	38	39
Total valid modifications	79	80	81
Average number of modifications per non-theorem	0.8	1.3	3.1
Average time to generate modifications (s)	73	120	253

Table 1
Results from modification attempts on 98 non-theorems

it was found to be valid, i.e., not obviously just a restatement of the conjecture. Taking these into account, in addition to the modifications stating that the conjecture is true if and only if the algebra is trivial (a valid modification), TM produced proper modifications for 79, 80 and 81 of the 98 of the non-theorems respectively, i.e., 81%, 82% and 83%. We believe that such a success rate is very encouraging. These figures don't appear to provide much evidence of improvement by running HR for longer and allowing it to use information from equivalence conjectures. However, if we look at the average number of modifications produced in the three sessions, we see that using the setup as in the first session, on average TM will find 0.8 proved modifications, but using the setup as in the third session, TM will find 3.1 modifications per theorem. However, the time taken to produce these modifications triples.

To illustrate why TM highlights theorems for which $M \Rightarrow C$, we can look at the non-theorem we generated from TPTP theorem GRP001. This states that, if all elements in the group square to give the identity, then the group must be Abelian, i.e., $(\forall\ a\ (a*a = id)) \Rightarrow (\forall\ a, b\ (a*b = b*a))$. We removed the inverse and associativity axioms to make this into a non-theorem. TM found only two specialisations to perform the modification, both of which were cautioned. The first was: $\nexists\ b, c, d\ (b*c = d \wedge c*b \neq d)$. This is obviously the specialisation into Abelian groups, hence, including this specialisation into a modified theorem produced: "in Abelian groups, if all elements square to give the identity, then the group is Abelian", which is trivially true. Hence TM was right in this case to caution us about this theorem.

In contrast, however, when we gave TM the non-theorem which we generated from TPTP theorem GRP011-4, it produced specialisations which were not at all obvious, and hence made interesting modifications. GRP011-4 is the left cancellation law, i.e., $\forall\ a, b, c\ ((a*b = a*c) \Rightarrow b = c))$. We took out the identity and inverse axioms to generate a non-theorem, and one of the five (cautioned) specialisations was: $\nexists\ b, c, d\ (b*c = d \wedge b*d \neq c)$ Hence the

modified theorem states that, in algebras for which $\forall\ x, y\ (x * (x * y) = y))$, the left cancellation law holds (with no mention of associativity).

TM managed to find valid modifications for 7 out of the 9 non-theorems that we took directly from the TPTP library, and these provide interesting illustrative examples of TM working as it should do. Firstly, TM successfully modified 3 out of 5 non-theorems in combinatory logic – a new domain for HR. For example, non-theorem COL073-1 states that, given certain axioms: $\forall\ y\ ((apply(y, f(y))) = (apply(f(y), apply(y, f(y)))))$. TM found a single specialisation from the 7 supporting examples Mace provided:
$\nexists\ b, c, d\ (apply(b, c) = d \wedge apply(b, d) \neq c)$. However, this was only true of the trivial algebra, and while Otter couldn't prove an equivalence between this specialisation and the trivial algebra, we cannot rule it out.

In group theory, the first non-theorem in the library is GRP024-4, which states that, given the definition of the commutator operator on two elements x and y being $comm(x, y) = x * y * x^{-1} * y^{-1}$, then this operator is associative if and only if the product of the commutator is always in the centre of the group (defined to be the set of elements which commute with all others). Hence this theorem states that: $\forall\ x, y, z\ (comm(comm(x, y)), z) = comm(x, comm(y, x)) \Leftrightarrow \forall\ u, v, w\ (comm(u, v) * w = w * comm(u, v)))$. Mace could not find any counterexamples to this, but it did find four groups for which the conjecture is true. As strategic withdrawal doesn't need any counterexamples, TM could continue. It found that, with the extra axiom that the groups are self inverse (i.e., $\forall\ x\ (x = x^{-1})$), the conjecture actually holds.

The first of two ring theory non-theorems taken directly from the TPTP library was RNG007-5, which states that, given a ring for which $\forall\ x\ (x*x = x)$, then $\forall\ x\ (x * x = id)$. Given the first property as an axiom, TM proved that the second property is equivalent to being the trivial algebra, which gives good justification for implementing this functionality. The second ring theory non-theorem was RNG031-6, which states that the following property, P, holds for all rings: $\forall\ w, x\ ((((w * w) * x) * (w * w)) = id)$ where id is the additive identity element. Mace found 7 supporting examples for this, and 6 falsifying examples. HR produced a single specialisation concept which was true of 3 supporting examples: $\nexists\ b, c\ (b * b = c \wedge b + b \neq c)$. Otter then proved that P holds in rings for which HR's invented property holds. Hence, while TM couldn't prove the original theorem, it did prove that, in rings for which $\forall\ x\ (x*x = x+x)$, property P holds. The specialisation here has an appealing symmetry. A proof of the modified theorem is given in the appendix.

5 Conclusions and Further Work

We have described and demonstrated the effectiveness of the TM automated theorem modification system. This is based on an implementation of methods prescribed in Lakatos's philosophy of mathematics, and relies on the interaction of the HR, Otter and Mace programs. In tests, TM modified 7 out of 9 non-theorems from the TPTP library into interesting, proved alternatives, and on an artificial set of 98 non-theorems, it produced meaningful modifications 80% of the time, which we believe is highly encouraging given that this is only the first version of the software. We intend to improve the implementation in at least the following ways:

• enabling it to strengthen modifications after it has weakened the original conjecture. For instance, if it has proved $A \Rightarrow (P \Leftrightarrow Q)$, try $A \Rightarrow (P \wedge Q)$. We expect this to result in more interesting theorems;

• extending the domains on which it works, to security protocols, chemistry and bioinformatics as well as other mathematical domains;

• automatically evaluating the modifications further, to enable TM to recognise interesting modifications from all those produced. For instance it might consider aspects of the proof, such as its length;

• using a failed proof attempt to suggest modifications to a conjecture C. This is Lakatos's method of *lemma incorporation*: given a counterexample to a conjecture, find which step of the proof it violates and then modify the conjecture by making that step a condition. The modified conjecture therefore becomes: $\forall\ x$ which satisfy proof step i, C holds. We are currently implementing this method;

• exploring the possibilities of using TM to suggest case splits for difficult but true theorems. For instance, given a theorem: $\forall\ x\ P(x) \Rightarrow Q(x)$, and a concept $C(x)$ which covers all the supporting examples and no counterexamples, then TM would form and attempt to prove (i) $\forall\ x\ (C(x) \wedge P(x)) \Rightarrow Q(x)$ and (ii) $\forall\ x\ (\neg C(x) \wedge P(x)) \Rightarrow Q(x)$. This suggests ways of automatically rephrasing a conjecture statement into one which can be proved.

TM is part of a larger project, in which we are implementing all of the methods prescribed by Lakatos in [9]. The aim of this project is to (a) provide a computational model for the use of Lakatos's ideas and (b) enhance the model and implementation of automated theory formation (ATF) as described in [1]. Our model of Lakatos-enhanced theory formation has developed along two axes: the sophistication of the conjecture-correcting methods which Lakatos

proposed, and the social nature of the discourse he described.

There are various reasons to automate theories of scientific discovery, including developing new techniques which aid scientists in their work [10]. We have demonstrated a new technique, namely automated theorem modification, which has the potential to aid mathematicians, by adding more robustness and flexibility to automated theorem proving. We believe that such robustness – in the case of TM, gained by the integration of deductive, inductive and model based techniques – will play an important part in the next generation of automated theorem provers.

Acknowledgements

We would like to thank Alan Smaill and John Lee for their continued input to this project. Special thanks to Bill McCune and Geoff Sutcliffe for supplying data and for their input to this work, and to Roy McCasland for providing the proof found in the appendix. We are grateful to the anonymous referees for their useful comments on an earlier draft of this document, and to the organisers of the IJCAR workshop on disproving. This work has been supported by EPSRC platform grant GR/S01771/01.

References

[1] S Colton. *Automated Theory Formation in Pure Mathematics*. Springer-Verlag, 2002.

[2] S Colton. The HR program for theorem generation. In *Proceedings of CADE*, 2002.

[3] S Colton and S Muggleton. ILP for mathematical discovery. In *Proceedings of the 13th International Conference on Inductive Logic Programming*, 2003.

[4] S Colton, A Bundy, and T Walsh. Automatic identification of mathematical concepts. In *Machine Learning: Proceedings of the 17th International Conference*, 2000.

[5] S Colton, A Bundy, and T Walsh. Automatic invention of integer sequences. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence*, 2000.

[6] 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.

[7] S Colton and I Miguel. Constraint generation via automated theory formation. In *Proceedings of CP-01*, 2001.

[8] S Colton and G Sutcliffe. Automatic generation of benchmark problems for automated theorem proving systems. In *Proceedings of the Seventh AI and Maths Symposium*, 2002.

[9] I Lakatos. *Proofs and Refutations: The logic of mathematical discovery*. Cambridge University Press, 1976.

[10] P. Langley. Lessons for the computational discovery of scientific knowledge. In *Proceedings of First International Workshop on Data Mining Lessons Learned*, 2002.

[11] W McCune. The OTTER user's guide. Technical Report ANL/90/9, Argonne National Labs, 1990.

[12] W McCune. A Davis-Putnam program and its application to finite first-order model search. Technical Report ANL/MCS-TM-194, Argonne National Laboratories, 1994.

[13] W McCune and R Padmanabhan. *Automated Deduction in Equational Logic and Cubic Curves, LNAI 1095*. Springer-Verlag, 1996.

[14] A Meier, V Sorge, and S. Colton. Employing theory formation to guide proof planning. In *Proceedings of the Tenth Symposium on the Integration of Symbolic Computation and Mechanized Reasoning, LNAI 2385*. Springer, 2002.

[15] K Popper. *The Logic of Scientific Discovery*. Basic Books, 1959.

[16] G. Sutcliffe and C.B. Suttner. The TPTP Problem Library: CNF Release v1.2.1. *Journal of Automated Reasoning*, 21(2):177–203, 1998.

[17] J Zimmer, A Franke, S Colton, and G Sutcliffe. Integrating HR and tptp2x into MathWeb to compare automated theorem provers. In *Proceedings of the CADE'02 Workshop on Problems and Problem sets*, 2002.

Appendix

Theorem:
Let R be a ring such that $\forall\, x \in R, x + x = x * x$. Then $\forall\, x, y \in R, x^2yx^2 = id$.

Proof. Let r be an arbitrary element in R. Then

$$-(r^2) = -(r * r) = -(r + r) = (-r) + (-r) = (-r) * (-r) = (-r)^2 = r^2$$

Hence $-(r^2) = r^2$, so $r^2 + r^2 = id$.

Now let x and y be arbitrary elements in R. Then:

$$\begin{aligned} x^2yx^2 &= (x^2y)(x^2) = ((x + x)y)(x^2) \\ &= (xy + xy)(x + x) = (xyx + xyx) + (xyx + xyx) \\ &= (xyx * xyx) + (xyx * xyx) = (xyx)^2 + (xyx)^2 \\ &= id \text{ [by above result]} \end{aligned}$$

□