Lakatos-style Automated Theorem Modification

Simon Colton[1] and **Alison Pease**[2]

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 through an interaction of the HR machine learning system, the Otter theorem prover and the Mace model generator, and uses methods inspired by Lakatos's philosophy of mathematics. We demonstrate the effectiveness of this approach by modifying non-theorems taken from the TPTP library of first order theorems.

1 Background

Current automated deduction systems are only capable of three types of output when given a conjecture: proof that it is true, proof that it is false, or that the conjecture is still open. We believe that greater flexibility and ability to better handle ill-specified problems would improve reasoning systems. We have implemented a theorem modifier system, TM, which is able to take in a conjecture, try to prove it and if unsuccessful (either because the conjecture is too hard to prove or because it is false), produce modified versions of the conjecture which it *can* prove. For instance, given the non-theorem that all groups are Abelian, TM states that it cannot prove the original result, but it has discovered that all *self-inverse* groups are Abelian.

TM is inspired by methods from [2]. Lakatos argued that mathematics developed in a much more organic way than its rigid textbook presentation of definition-theorem-proof would suggest. He saw mathematics as a process in which, via patterns of analysis which he categorised into seven methods, conjectures and proofs are gradually refined but never certain. In [2], Lakatos 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. The two methods we have drawn on for TM are Lakatos's exception-barring methods: *piecemeal exclusion* and *strategic withdrawal*. The former works by generalising from a counterexample to a class of counterexamples and then excluding this class from the faulty conjecture; and the latter 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. Put formally, given the conjecture $\forall\ x\ P(x) \Rightarrow Q(x)$, counterexamples n such that $P(n) \wedge \neg Q(n)$, and positive examples p such that $P(p) \wedge Q(p)$:

• *piecemeal exclusion* finds a concept C such that $\forall\ n\ C(n)$, and $\forall\ p\ \neg C(p)$, then modifies the conjecture to:
$\forall\ x\ (\neg C(x) \wedge P(x)) \Rightarrow Q(x)$

• *strategic withdrawal* finds a concept C such that $\forall\ p\ C(p)$, and $\forall\ n\ \neg C(n)$, then modifies the conjecture to:
$\forall\ x\ (C(x) \wedge P(x)) \Rightarrow Q(x)$.

[1] Department of Computing, Imperial College, London, United Kingdom; email: sgc@doc.ic.ac.uk
[2] School of Informatics, University of Edinburgh, United Kingdom; email: alisonp@dai.ed.ac.uk

In order to automate these methods we need a program which attempts to prove a conjecture (we do not want to modify conjectures which can be proven automatically); one which generates supporting and counterexamples, and one which generates concepts to cover a set of entities. To this end TM interacts with the Otter theorem prover [3], the Mace model generator [4], and the HR machine learning system [1]. Otter is a first order resolution theorem prover which has been used for many discovery tasks in algebraic domains, e.g., [5]. Mace is a model generator which employs the Davis-Putnam method for generating models to first order sentences. HR takes in objects of interest, such as groups, and background concepts such as the operator of a group, and forms a theory containing concepts, conjectures and proofs. Its concept formation functionality works by applying one of 15 production rules to one (or two) old concepts to generate a new concept. For instance, it might pass the concept element through the *size* production rule to produce the function $f(a) = |\{x : x\ \in\ G\}|$, i.e. the size of a group.

2 The Theorem Modifier System

TM works by taking in a conjecture of the form $A \Rightarrow C$ where A is a conjoined set of axioms, and C the conjecture a user wishes to prove/modify/disprove. We have so far limited TM to algebraic domains, such as group theory, where there is a single operator which satisfies the axioms of associativity, identity, and inverse.

Stage 1: Given a conjecture, TM first performs two preliminary checks to see whether it is worth modifying. These are:
(i) if Otter can prove in a user-specified period of time that the conjecture is true, i.e., $A \Rightarrow C$, then TM reports this and returns the proof;
(ii) TM negates the conjecture and invokes Otter to try to prove the negation.

Stage 2: If unsuccessful, then TM performs a further check before invoking MACE and HR – whether the conjecture is true if and only if we limit the objects in the domain to (a) the trivial algebra, i.e. if $A \Rightarrow ((\forall\ a, b\ (a = b)) \Leftrightarrow C)$, or (b) non-trivial algebras, i.e. if $A \Rightarrow ((\exists\ a, b\ (a \neq b))) \Leftrightarrow C$. This check is a type of modification inspired by Lakatos's exception-barring methods, where (a) the only supporting example is the trivial algebra, so we limit the domain to this (a type of strategic withdrawal), and (b) the only counterexample is the trivial algebra, so we exclude it form the conjecture (a type of piecemeal withdrawal). We apply these methods separately at this stage as it is often the case that a theorem is true *only* for the trivial algebra, in which case the theorem is usually uninteresting. The opposite case, that the theorem is true for everything *but* the trivial algebra, is rare.

Stage 3: If the conjecture gets beyond these checks, then TM has the chance to modify it. To do this;
(i) TM invokes MACE to generate two sets of algebras; the first containing those which support the conjecture, and the second those which contradict it.
(ii) These sets are then passed to HR as objects of interest, as well as the conjecture, from which it extracts the core concepts. It uses this input to produce a theory, for a user-specified number of steps.
(iii) TM then identifies all the specialisations of the algebra (such as Abelian or self-inverse algebras) which HR has invented. TM extracts those which describe only the algebras which support the conjecture (or a subset of them). For each extracted specialisation, M, TM forms the modified conjecture $(A \wedge M) \Rightarrow C$ by adding M to the axioms.
(iv) Otter is invoked to see which of these modifications can be proved, and any which are proved are presented to the user.

Stage 4: Finally, TM evaluates whether its modifications are likely to be interesting to the user. It does this by testing whether:
(i) the only example to satisfy M is the trivial algebra, in which case TM invokes Otter to check whether $A \Rightarrow (M \Leftrightarrow (\forall\ a, b\ (a = b)))$;
(ii) the concept is a redefinition of the conjecture statement, for instance, M is the condition that a group is Abelian, when the original conjecture was *all groups are Abelian*, i.e. the modification is that *all Abelian groups are Abelian*. If *every* supporting example has the property prescribed by M, then TM uses Otter to try to prove: (a) $M \Leftrightarrow C$, (b) $A \Rightarrow (M \Leftrightarrow C)$, (c) $M \Rightarrow C$. TM marks these as probably uninteresting, though still reports them to the user, as it may be that the equivalence of C and M is surprising and non-trivial, and hence the modified conjecture interesting.

Note that this process of modifying conjectures by specialising them is an implementation of Lakatos's strategic withdrawal method. However since TM instructs HR to use its negate production rule, 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. Therefore TM also uses piecemeal exclusion to form the modifications.

3 Testing and Evaluation

We have tested the hypothesis that TM can find meaningful modifications to non-theorems by using the TPTP library [6]. From theorems in the group (GRP), field (FLD), ring (RNG) and combinatory logic (COL) domains in this library we generated 89 non-theorems by removing axioms, changing/removing quantifiers, altering variables and constants, and altering bracketing. To this set we added 9 of the non-theorems in TPTP to make a test set of 98 non-theorems. In addition to testing the hypothesis, we investigated how we could improve performance - where performance was measured by *(a)* the number of non-theorems for which TM was able to find an interesting modification, and *(b)* the number of interesting modifications produced per non-theorem. We considered a modification to be interesting if *(i)* TM evaulated it as interesting in stage 4, or *(ii)* it showed that the orginal conjecture is true only if the algebra is trivial, or *(iii)* it comes with the caution that the specialisation may trivially make the theorem true, but is not an obviousl restatement of the conjecture.

In order to investigate how performance could be improved, we altered the amount of time Mace, Otter and HR were each allowed. Mace only found a few more examples with extra time, which did not affect the specialising concepts that HR found. Similarly giving Otter more time did not improve the performance of TM – giving it extra time did not result in it proving a difficult problem. Therefore, 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 find specialisations was turned off [1].

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

TM produced interesting modifications for 79, 80 and 81 out of 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 found 0.8 proved modifications per non-theorem, but using the setup as in the third session, it found 3.1 modifications per non-theorem. The drawback is that the time taken to produce these modifications triples.

As an illustrative example, a non-theorem from ring theory (RNG031-6) 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.

The TM system demonstrates increased flexibility in automated theorem proving, gained by integrating deductive, inductive and model based techniques. We believe that such flexibility will play an important part in the next generation of automated reasoning sytems, and that the combination of areas such as machine learning, constraint solving and theorem proving is inevitable for AI to progress.

REFERENCES

[1] S Colton. *Automated Theory Formation in Pure Mathematics.* Springer-Verlag, 2002.
[2] I Lakatos. *Proofs and Refutations: The logic of mathematical discovery*. Cambridge University Press, 1976.
[3] W McCune. The OTTER user's guide. Technical Report ANL/90/9, Argonne National Laboratories (ANL), 1990.
[4] W McCune. A Davis-Putnam program and its application to finite first-order model search. Technical Report ANL/MCS-TM-194, ANL, 1994.
[5] W McCune and R Padmanabhan. *Automated Deduction in Equational Logic and Cubic Curves, LNAI 1095*. Springer-Verlag, 1996.
[6] G Sutcliffe and C Suttner. The TPTP problem library: CNF release v1.2.1. *Journal of Automated Reasoning*, 21(2):177–203, 1998.