Automatic Conjecture Modification

Alison Pease* and Simon Colton**
*School of Informatics, University of Edinburgh, UK. alisonp@dai.ed.ac.uk
**Department of Computing, Imperial College London. sgc@doc.ic.ac.uk

One of the original goals of writing theorem provers was to provide tools which mathematicians could use to aid their research, yet this goal remains unsatisfied to a large extent. We believe that one of the reasons for this failure is that so far automated theorem provers have failed to incorporate the flexibility which humans take for granted, and which is inherent in intelligent activity. To begin to address this, we have implemented a theorem modification system, called TM. This 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. As a simple, yet illustrative, example, 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 a *combined reasoning* system, which uses inductive, deductive and model-based reasoning to achieve the goal of modifying theorems.

TM is inspired by methods suggested in Imre Lakatos' philosophy of mathematics [4]. Lakatos argued that mathematics developed in a much more organic way than the rigid textbook presentation of definition-theorem-proof would suggest. He saw mathematics as an adventure in which – via patterns of analysis which he categorised into seven methods – conjectures and proofs are gradually refined but never certain. 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. Two of Lakatos' methods which have inspired our implementation of TM embed notions of exception-barring, namely *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 $P(p) \wedge Q(p)$, then:

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

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

In order to perform such modifications, TM needs four basic abilities: (a) to prove conjectures (b) to generate examples to support and contradict a conjecture (c) to generate concepts which describe sets of examples and (d) to use these concepts in piecemeal exclusion and strategic withdrawal modifications. TM takes care of the last of these abilities, but relies on third party software to achieve the first three. In particular, as described below, TM interacts with the Otter resolution theorem prover [5], in order to prove theorems, and the Mace model generator [6] to find examples. It also uses the HR machine learning system [1] to generate concepts which describe sets of examples [2].

We describe how TM works in algebraic domains such as group theory and ring theory, but this approach extends to non-algebraic domains. TM is given – in Otter syntax – 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. It then develops the conjecture via the following four stages:

Stage 1: TM first performs two preliminary checks to see whether the conjecture needs modifying and can be modified. If (i) Otter can prove in a user-specified period of time that the conjecture is true, then TM reports this and returns the proof. However, if (ii) TM negates the conjecture and Otter prove the negation theorem then neither strategic withdrawal nor piecemeal exclusion – both of which essentially perform specialisations – will rectify the situation, hence TM stops.

Stage 2: If neither (i) nor (ii) in stage 1 succeeds, then TM performs a further check before proceeding, namely 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 from 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 *except* the trivial algebra is rare, but not impossible, e.g., in group theory, non-trivial groups have a non-idempotent element (for which $a * a \neq a$), which is a property not true of the trivial group.

Stage 3: If the conjecture gets beyond these checks, then TM has the chance to modify it. This is achieved in the following way: (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, which extracts the models from Mace's output in addition to the core concepts in the domain (e.g., multiplication, identity, inverse, etc.) HR uses this input to produce a theory of the domain, and is allowed a user-specified amount of time to do this. (iii) TM then identifies all the different types of group (such as Abelian or self-inverse groups) 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.

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 separates those modified conjectures which satisfy (i) or (ii) from those which don't. It marks the former modifications as possibly uninteresting, but does not discard them, because, in our experience, often the equivalence of M and C, or M and the trivial algebra, is surprising and non-trivial, and hence the modified conjecture is 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 (see [2], 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.

We have tested the hypothesis that TM can find meaningful modifications to non-theorems by using the TPTP library [7]. From group theory, ring theory, field theory and combinatory logic theorems in this library, we artificially generated non-theorems by removing axioms, changing/removing quantifiers, altering variables and constants, and altering bracketing. Setting Otter and Mace to run for 10 seconds and HR for 3000 theory formation steps, given 91 non-theorems, TM produced valid modifications for 83% of them, and found an average of 3.1 modifications per non-theorem. These results are given in [3], where we also report that increasing Otter and Mace's time limit has little effect on the abilities of TM, but increasing HR's time limit produces a limited improvement.

We also tested TM on the 9 suitable TPTP non-theorems that we could find, and it found modifications to 7 of them (78%). For instance, we gave TM non-theorem number RNG031-6 from the library, namely 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, and HR produced a single specialisation concept which was true of 3 supporting examples. This led to the following modified theorem which Otter proved: in rings for which $\forall\, x\ (x * x = x \Rightarrow x + x = x)$, property P holds. Hence, while TM couldn't prove the original conjecture (hardly surprising as it was false), it supply an interesting, proved, modified theorem which was based on the original.

We believe that the ability to modify as well as prove and disprove theorems constitutes an important new step in the development of automated theorem provers. By modelling some of the ways in which human mathematicians work – as prescribed by Lakatos – we have built a system which we hope will eventually enable provers to be more useful to human mathematicians.

References

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

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

[3] S Colton and A Pease. Lakatos-style automated theorem modification. *Submitted to ECAI'04*, 2004.

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

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

[6] 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.

[7] G Sutcliffe and C Suttner. The TPTP problem library: CNF release v1.2.1. *Journal of Automated Reasoning*, 21(2):177–203, 1998.