Lakatos-style Methods in Automated Reasoning

Simon Colton* **and Alison Pease****

* Department of Computing, Imperial College London, UK
** School of Informatics, University of Edinburgh, UK

Abstract

We advocate increased flexibility in automated reasoning, whereby a reasoning agent is able to correct the statement of a given faulty conjecture in order to prove that the modified theorem is true. Such alterations are common in mathematics. In particular, in his book 'Proofs and Refutations', Imre Lakatos prescribes various techniques for the modification of a faulty conjecture within a social setting (a hypothesised mathematics class). This has inspired a multi-agent approach to automating Lakatos-style techniques, and we give details of the implementation of these methods within (and on top of) the HR automated theory formation system. We report on the progress of this project and supply illustrative results from sessions using the enhanced system.

1 Introduction

Because of the polished nature of results presented in textbooks, it could be perceived that mathematics proceeds via the proposition of a perfectly formed theorem statement followed by an equally perfect proof of the truth of the statement. However, this is a mistake: it is more common that a sketch conjecture is outlined and while a proof for the result is sought, the conjecture statement evolves into a result which can actually be proved. Following this will be a rationalisation of the theory surrounding the result, whereby concept definitions are stated externally and lemmas are extracted in order to make the theorem statement and proof of it as comprehensible (and beautiful) as possible.

After abortive attempts to settle an open conjecture, a human mathematician has (at least) two further options: to prove that a conjecture is independent of the axioms, i.e., cannot be proven or disproven, or to modify it to something that *can* be proved. The latter option can also be taken if a counterexample has been discovered which proves the conjecture false, and yet the result still appears to be interesting and worth pursuing (modifying). Well known examples where the first option was taken include:

- The continuum hypothesis – there is no number which is larger than the size of the set of natural numbers and less than the size of the set of real numbers.

- The axiom of choice – if S is a set of disjoint non-empty sets then there exists at least one set which contains a single member from each member of S.

- The parallel postulate – suppose a straight line falling on two straight lines makes the interior angles on the same side less than two right angles. Then the two straight lines, if produced indefinitely, will meet on the side where the angles are less than the two right angles.

Well known examples where the second option was taken include:

- Euler's conjecture – for all polyhedra, the number of vertices (V) minus the number of edges (E) plus the number of faces (F) is 2, which was modified to various conjectures, including: for all convex polyhedra, V-E+F=2; for all simply-connected polyhedra with simply-connected faces, V-E+F=2; and for all polyhedra whose circuits bound, V-E+F=2 [12].

- The principle of continuity – the limit of any convergent series of continuous functions is itself continuous, which was modified to: a uniformly convergent series of continuous functions has a continuous limit [12].

- The conjecture that all even perfect numbers end *alternately* in 6 or 8, which was modified to all even perfect numbers end in *either* 6 or 8 [2].

Given a conjecture, most automated reasoning programs would currently try to prove it true (e.g., Otter, [13]) – resulting in a theorem, or false (e.g., MACE, [14]) – resulting in a non-theorem. If they are unsuccessful within a specified time period, then the conjecture remains open.

We believe that the kind of flexibility as exhibited in the two options above will be an essential part of the next generation of automated theorem provers. It has often been lamented that theorem proving systems have not captured the attention and imagination of research mathematicians. There are many possible reasons for

this: (i) overuse of formal notation (ii) poor proving power of programs (iii) difficulty to use or (iv) unreliability of programs. Another possible reason is the flexibility problem: it may be that a mathematician is (a) not sure whether the conjecture as stated is true and (b) not sure whether the conjecture as stated is exactly how they intended it to be.

As an illustrative example, imagine a young child experimenting with prime numbers. If they asked an automated theorem prover to show that all primes were odd, they would be told that this was false. It would be much more educational if the theorem prover stated that they were nearly right: all primes *except two* are odd. This was possibly the hypothesis that the child had intended, and the proof of this result may inspire greater understanding and exploration.

The first option above, namely showing that a conjecture is independent of the axioms is beyond the scope of this paper, and we focus here on the second option outlined above. In particular, we discuss processes by which a faulty conjecture can be analysed to produce a modified conjecture (and proof of it). These methods are inspired by the insights of Imre Lakatos as expressed in [12]. Using Euler's conjecture (see above) as a running example, and employing a setting whereby a hypothesised group of students and a teacher discuss the result, Lakatos proposes many possible ways to fix the faulty theorem statement.

The main purposes of the project described in this paper are (a) to provide a computational model for the use of Lakatos's ideas and (b) to enhance the model and implementation of automated theory formation (ATF) as described in [4]. In §1.1, we briefly describe how the work fits into the notion of automated theory formation. The model of Lakatos-enhanced theory formation has developed along two axes. To model the social nature of the discourse proposed by Lakatos, we have implemented a multi-agent system to propose, criticise and ultimately fix faulty conjectures arising in an ATF setting. The first axis of development is therefore the sophistication of the agent interaction, as described in §2. The second axis is in terms of the sophistication of the conjecture-correcting methods proposed by Lakatos, as described in §3.

In §4 we present some illustrative examples of the enhanced system making and attempting to fix some faulty conjectures. We conclude by highlighting our current progress with respect to the two axes mentioned above, and by suggesting some possible applications of the completed system to more specialised reasoning domains such as automated theorem proving, machine learning and constraint solving.

1.1 Automated Theory Formation

Automated Theory Formation, as advocated in [4], aims to combine various reasoning techniques (inductive, deductive, abductive, etc.) in order to build theories containing concepts, hypotheses, examples, proofs, etc. The theories were originally constructed only in mathematical domains, and built from first principles, e.g., axioms and/or basic concepts supplied with objects of interest in the domain (integers in number theory, groups in group theory, etc.) However, recently, HR has been applied in other scientific domains, starting with more sophisticated background knowledge [3].

The functionality behind HR is based on a set of 12 production rules which invent new concepts based on old concepts [7]. The search for concepts is heuristic: HR has many measures of interestingness for concepts and it takes a weighted sum of these measures in order to determine an overall evaluation of each concept. It builds new concepts from the more interesting old concepts before the less interesting ones.

As the invention process proceeds, patterns are sought in the examples of the concepts, and various hypotheses are made, for instance that the examples of one concept are all examples of another concept, which leads to the statement of an implication conjecture. Similar equivalence, non-existence and applicability conjectures are made using empirical evidence, as described in [5]. In mathematical domains, where axioms have been supplied by the user, attempts to settle the conjectures are also made. In particular, HR employs the Otter theorem prover [13] to attempt to prove theorems, and the MACE model generator [14] to attempt to find counterexamples to non-theorems.

Of importance to the project described here, we have recently added more flexibility to HR's conjecture making abilities. In particular, it is now able to produce 'near' conjectures, such as *near-equivalence* conjectures which state that one concept has the same examples as another concept with a few exceptions. Similarly, HR can make *near-implication* conjectures, which state that all the examples of one concept are examples of another concept, again with a few exceptions. In both cases, the user can set a limit for the proportion of objects of interest in the domain which can be exceptions. Note that we write: $A \rightsquigarrow B$ for the near conjecture that concept A implies concept B. Similarly, we write: $A \leftrightsquigarrow B$ for near equivalences.

Also of importance to this project is the notion of 'forcing' concepts. This is a process whereby the user can guide the theory formation process in terms of which production rule steps to carry out in which order. This can be done interactively (on-screen) or via a file of instructions. Usually this guidance will be towards the construction of particular concepts. Therefore, if the guidance is undertaken at the very start of a theory formation session, we can see forcing steps simply as a different way of providing background information to the system. This is because forcing concepts guarantees their existence at the start of the theory in the same way as explicitly providing them.

The extra implementation for this project has been (i) to add Lakatos-inspired techniques (see §3) to the main functionality of HR, so that it can correct faulty conjectures, as described in §3 and (ii) to build a multi-agent system based on the teacher-student protocol suggested by Lakatos, in such a way that each agent has access

to the full range of abilities within HR, as discussed in §2. We have previously showed in [6] that a multi-agent model for theory formation (with little of the sophistication of the Lakatos-enhanced system described here), can qualitatively improve the search undertaken by the system as a whole. In addition to providing a computational model for Lakatos's ideas and possible applications to other areas of Artificial Intelligence, we believe the enhancement of the model of theory formation via a multi-agent realisation of Lakatos's methods will substantially improve the ability of the implemented system.

It is obvious that using Lakatos inspired techniques will add to the richness of concepts and conjectures that HR can make, because it can now make false (near) conjectures and fix them in such a way as to make them true. Moreover, we believe the agent architecture will increase both the efficiency and creativity of the system as a whole. In terms of the efficiency, as with many multi-agent systems, we will be able to distribute processes in an intelligent way over networks. In terms of the creativity of the system, we believe that creative actions may follow from mistakes made by individual agents, via argumentation and other forms of discourse within the group. In fact, we argue in [6] that a simple multi-agent architecture added to the creativity of the system as a whole. We intend to test these improvements with substantial experimentation, once the final version is ready.

Given the main application of this work to the improvement of automated theory formation, we describe both the agency and the Lakatos methods implemented in terms of improved automated theory formation. In §5, we return to the question of other potential applications of this work.

1.2 Related Work

To our knowledge, our project is the first to use an agency to discover and correct faulty conjectures via a voting mechanism dependent on methods proposed by an expert (Lakatos). Various other projects are related to this. In particular, finding counterexamples to non-theorems is performed by model generators such as MACE [14]. Also, in chapter 11 of [1], Bundy describes the productive use of failure within theorem proving attempts, e.g., he advocates the alteration of induction schema in order to generalise a theorem to be proved. Fixing of faulty conjectures has also been studied in [15]. They use proof planning to reduce the search for antecedents, which, when added to an axiom system will non-trivially make the theorem true. Their application was to theorems proved inductively, and their system correctly modified 80% of 45 non-theorems about integers and lists, e.g., the non-theorem that $half(x) < x$, was altered to: $x \neq 0 \rightarrow half(x) < x$.

Various approaches to agent-based reasoning have also been studied. In particular, Delgrande and Mylopolous propose logics for making decisions in a committee [10]. Also, approaches to general problem solving via distributed deduction are discussed in [11].

2 Levels of Agent Sophistication

In order to simulate the discussion in [12] we are implementing the methods within an agent architecture, consisting of a number of students and a teacher such that each agent has a copy of HR. Our system is written in Java and is distributed over different machines, with the agents using sockets to communicate. The number of agents is flexible, determined by the user, but in each case, one agent is meant to represent a teacher, with the other agents being students.

The *problem* of the agency is to model the social process of concept and conjecture refinement described by Lakatos, with the *task* being to develop interesting concepts and conjectures to add to the theory. The *knowledge* the agency starts with is the input to each copy of HR, which consists of objects of interest, background concepts and possibly concepts which have been forced by the user. The *motivation* of the students is to accept, modify, or reject a conjecture, and this is done by *actions* which are Lakatos's methods. The students *communicate* by sending conjectures, concepts, and counterexamples, and the teacher by sending requests such as: work independently; send a concept to cover counterexamples $[x, y, z]$; or send a modification to faulty conjecture C. *Discussion* is directed by the teacher who keeps a group agenda and adds responses to it – in a depth, breadth or best-first manner. Items which have been discussed are recorded in a discussion vector.

The global theory is the collection of conjectures, concepts, objects of interest (both those provided by the user and those discovered as counterexamples to conjectures), and proofs which the group discuss and accept. Three main factors can influence the production of the global theory. These are the user, the teacher and the students, and the complexity of the system increases respectively with the degree of influence that these have. We distinguish the different layers of complexity in the interaction between agents as follows:

1. Maximum user input (puppet show). The user sets flags to control which methods can be used by the students, and what should be added to the group agenda (and in which order).
2. Teacher/user employing students as references (dictatorship), i.e., the teacher plays the role of the user above.
3. Teacher allowing students to make decisions – the students decide which method they want to use on a particular faulty conjecture.
4. Students decide the group agenda and global theory (democracy). Students decide about the order in which responses are inserted into the agenda, i.e., they evaluate all responses and the order is decided by the group, for example by a vote.
5. Students allowed class discussion – they can interact with each other directly and are not tied to responding to the teacher's requests. They can also send requests themselves.

Currently, our agent architecture is at level 2. We are implementing abilities for the students to make their own decisions (level 3).

In terms of the BDI – beliefs, desires and intentions – approach outlined in [18], the agents' *beliefs* consist of the data input to their copy of HR. Their *desires* include building an interesting theory, accepting, modifying or rejecting a specific conjecture (depending on the proportion of their examples it holds for, and how interesting it is according to their evaluation function). Their *intentions* are to perform specific Lakatos methods or parts of the methods. For details of how the Lakatos's methods (in particular, monster-barring) fit into the literature on argumentation-based negotiation, see [17].

3 Lakatos's Methods

Lakatos identified many methods from mathematical practice in [12], which we briefly characterise below. Broadly speaking, the first of these is simply scientific induction, where hypotheses are produced by generalising from particular examples (as opposed to mathematical induction, which is a proof procedure). The second of these is surrender, whereby a decision is taken *not* to attempt to modify a faulty conjecture. The next five methods describe ways of fixing faulty conjectures in the light of known counterexamples. The final two methods – *lemma incorporation* and *proofs and refutations* – describe how counterexample finding and modification of theorem statements can be used to fruitfully prove mathematical results. Given a conjecture, C, Lakatos's methods can be described as follows:

1. Induction: generalise from particulars.

2. Surrender: look for counterexamples to C and use them to refute the conjecture.

3. Monster-barring: modify the definition of objects in the theory, to exclude an unwanted counterexample.

4. Piecemeal exclusion: find the properties which make the counterexamples break C, and then modify C by excluding objects with those properties.

5. Strategic withdrawal: consider the examples for which C does hold, generalise properties of those examples, and limit C to only examples with those properties.

6. Monster adjusting: reinterpret a counterexample so that it no longer violates C.

7. Lemma incorporation: given a *global* counterexample (i.e. one which violates C), find which step of the proof it violates and then modify C by making that step a condition. Given a *local* counterexample (which violates a proof step but not C), look for a hidden assumption in the proof step, then modify both the proof and C by making the assumption an explicit condition.

8. Proofs and refutations: use the proof steps to suggest counterexamples. For any counterexamples found, test whether they are local or global counterexamples and perform lemma incorporation.

3.1 Implementation Details

So far, we have implemented methods based on Lakatos's notions of surrender, piecemeal exclusion and strategic withdrawal. Note that there has been a certain amount of rationalisation and interpretation – within the context of automated theory formation – of the methods described by Lakatos. In particular, we have had to be more concrete about the application of the methods. For instance, the implemented techniques differ slightly when applied to a near-equivalence or a near-implication.

Note also that, with the exception of surrender, an agent would deploy one of these techniques by (a) determining what the nature of the fault in the conjecture is, e.g., finding the relevant counterexamples and identifying whether concepts on the left, right, or both sides of a near conjecture have to be altered, and (b) adding steps to the agent's version of HR's agenda which will effectively force the theory formation. The forcing of the theory formation steps will lead to the altered versions of concepts being introduced, which in turn will lead to an altered version of the original conjecture being discovered. Note that both the original and the modified conjecture will appear in the theory.

We explain the methods below in terms of conjectures which relate concepts of a binary nature. Binary concepts express properties of the objects of interest, such as whether an integer is prime or not, or whether a group is cyclic or not, etc. The advantage to this is that we can talk about those objects of interest which are *positive* with respect to a concept, for instance, we can say that 2,3,5,7 are positives for the concept of prime numbers. Likewise, we know that the integers 1,4,6,8,9 are *negatives* for the concept of prime numbers. Note that the methods generalise to deal with conjectures of any arity, but this functionality is not yet implemented.

• Surrender

This is the simplest method, as there is no conjecture refinement – the conjecture is simply abandoned in the face of a set of counterexamples. Until recently, this has been HR's only mode of operation. However, in the enhanced system, given a faulty conjecture, only if an agent is requested to perform surrender (either by the user or the teacher), or it is unable to find any suitable conjecture refinement, will an agent surrender the conjecture.

It is very important to know which conjectures to surrender and which to attempt to refine, because, in a theory formation setting, an agent may be generating or receiving large numbers of conjectures. We are currently investigating more sophisticated ways of measuring the interestingness of a conjecture which has been broken by counterexamples. For instance, if an ordering over the objects of interest is available (e.g., the natural ordering of the integers), then possibly more counterexamples should be allowed before surrendering, if the counterexamples are small – in the sense of the ordering. There are many examples of theorems to which numerous small (often trivial) objects are the only counterexamples.

• Concept-barring
Inspired by Lakatos's piecemeal exclusion methods, we have differentiated between concept-barring, which is where a concept is excluded from a conjecture statement, and counterexample-barring, where counterexamples are listed in the conjecture as exceptions (see below). In concept-barring, given a near equivalence $P \nleftrightarrow Q$, agents first determine whether the counterexamples are positives for P only, positives for Q only, or split between the two. If all counterexamples are positives for P, an agent will:

(i) Find a concept, X, in the theory which exactly covers the counterexamples. By this, we mean that the positives for X are exactly the counterexamples, with no exceptions.
(ii) Form the concept $P \wedge \neg X$ by forcing a 'negate' production rule step onto the agenda and carrying it out (see [4] for a description of the negate rule).
(iii) Make the conjecture $P \wedge \neg X \leftrightarrow Q$.

The agent will undertake a similar routine if all the counterexamples are positives for Q. If some counterexamples are positives for P and others for Q, the agent will:

(i) Find a concept, X, in the theory which exactly covers the counterexamples which are positive for P.
(ii) Form the concept $P \wedge \neg X$ via the appropriate forced negate step.
(iii) Find a concept, Y, in the theory which exactly covers the counterexamples which are positive for Q.
(iv) Form the concept $Q \wedge \neg Y$ via the appropriate forced negate step.
(v) Make the conjecture $P \wedge \neg X \leftrightarrow Q \wedge \neg Y$.

We intend to allow more flexibility to this method, by enabling the agents to find concepts which cover a superset of the counterexamples. The agent deals with near-implications similarly. In the case of an implication such as $P \rightsquigarrow Q$, the counterexamples must be positive for P and negative for Q, hence the agent will construct the conjecture $P \wedge \neg X \rightarrow Q$ using an appropriate concept X, if it can find one in its theory.

• Counterexample-barring
Given a faulty conjecture for which an agent cannot find concepts to cover the counterexamples, the agent will choose to perform counterexample-barring. Given a near implication $P \rightsquigarrow Q$, with counterexamples $[x, y, z]$, the agent will:

(i) Use the 'entity_disjunct' production rule to form the concept of an object of interest being either x, y or z. This is a new production rule implemented for this application to Lakatos's methods (such functionality can be replicated using HR's existing split and disjunct production rules, but implementing a new production rule was a neater solution). Hence the agent will form the concept, X, of objects of interest, n, for which

$$n = x \vee n = y \vee n = z.$$

(ii) Apply the negate production rule to P and X, to produce the concept $P \wedge \neg X$.
(iii) Make the conjecture $P \wedge \neg X \rightarrow Q$.

Note that, for both counterexample and concept-barring, it may be productive to split near-equivalences into two near-implications and apply the routine to both near-implications. For instance, starting with the faulty conjecture *x is prime* $\nleftrightarrow$ *x is odd*, an agent might produce the conjecture that primes except 2 are odd and the conjecture that odd non-squares are prime. In this case, the second conjecture turns out to be false, but in general, splitting near-equivalences into near-implications may be fruitful. This is analogous to how HR splits normal equivalences, as discussed in [5], and we are currently implementing it for near-equivalences.

• Strategic withdrawal
Strategic withdrawal aims to find a property of (some of) the objects of interest which *do not* break the conjecture, such that the property is true of none of the counterexamples. The modified conjecture statement would then restrict the scope of the original conjecture by declaring that only objects of interest with the property are to be considered. Naturally, this could be achieved by finding a property true of only the counterexamples and negating it, which basically implements the concept-barring method described above. However, in the general case, the property may be true of only a (preferably large) subset of the objects which do not break the conjecture. Hence we see that strategic withdrawal has the potential to modify conjectures in different ways to the concept-barring method. In addition, given that these methods operate within a multi-agent system, one agent (usually the teacher) can request a concept which covers certain objects, and then use the most interesting concept it receives in the conjecture refinement. Hence, using both strategic withdrawal and concept-barring give a larger choice of concepts from which to select.

Given a near-equivalence $P \nleftrightarrow Q$, the agent determines whether each counterexample is a positive for P or Q. If all the counterexamples are positives for P, then the agent:

(i) Finds a concept, X, in the theory which exactly covers the objects of interest which are positive for both P and Q (and is different from Q).
(ii) Makes the following conjectures: $X \leftrightarrow Q$; $X \rightarrow Q$; $X \rightarrow P$.

Similarly, if all the counterexamples are positives for Q, the agent forms the conjectures $Y \leftrightarrow P$; $Y \rightarrow P$; $Y \rightarrow Q$, for a suitable concept, Y, (which is different from P).

If some counterexamples are positives for P and others are positives for Q, then the agent:

(i) Finds a concept, Z, in the theory which exactly covers the positives for both P and Q.
(ii) makes the following conjectures: $Z \rightarrow P$; $Z \rightarrow Q$.

We intend to allow more flexibility in this method, by enabling the agents to find concepts which cover only a subset of the non-counterexamples.

Different methods may lead to the same conjecture refinement. For instance, we could rephrase the piecemeal exclusion example *all integers except squares have an even number of divisors* as an example of strategic withdrawal if we replace 'all integers except squares' with 'all non-squares' in the refined conjecture statement. Redundancy in HR's search space has always both (a) posed a problem, in terms of making it realise that it has invented the same concept twice and avoid making dull tautology conjectures and (b) been a source of power for the system: often, concepts we thought were outside of its search space have been re-invented with non-standard definitions.

4 Illustrative Results

We have been testing the development of our system with examples in number theory. This is a different domain to the domain of the running example in Lakatos's work, and thus enables us to test how general the methods are (Lakatos claimed that they were not specific to the polyhedra domain). We aim to validate the implemented techniques in different domains (including polyhedra) at a later date. We present below some runs which illustrate how the implemented methods work. Some of the sessions were contrived to produce particular – illustrative – results quickly, so that we did not have to sift through the output from the system to find the intended result. We hope to present results from less contrived sessions when the completed system is used for longer theory formation sessions at a later date.

In all cases, the interaction was via the teacher who asked the students to work independently for 20 theory formation steps. The teacher then put the conjectures it received onto the agenda in a depth-first manner and requested modifications from each student for each conjecture in turn. The students had individual ways of evaluating the conjectures they produced (for details of the measures of interestingness used in HR, see [8]). The specifics of the evaluation are not relevant here.

4.1 Concept-barring

We ran the agency with three students and a teacher. The first student started with the integers 1-10, the second with the integers 11-50 and the third with the integers 51-60. Each student started with the background concepts of integers, divisors and multiplication, and the forced concepts of squares and integers which have an even number of divisors. The teacher asked the students to send back their best equivalence conjecture, when evaluated using the weighted sum of measures of interestingness specific to the agent. The user set flags enabling all students to perform both concept-barring and counterexample-barring.

The third student formed the conjecture that x is an integer if and only if it has an even number of divisors, which is true of the integers 51-60. Students 1 and 2 found counterexamples 1, 4, 9 and 16, 25, 36, 49 respectively. Students 1 and 2 then looked for a concept to cover these and both found the concept of square numbers in their theory covered their counterexamples precisely (this is not surprising, as it was forced by the user at the start). They then formed the new concept of *non-squares*. In forming the new concept, they discovered the conjecture that x is non-square if and only if it has an even number of divisors. We see that the original equivalence conjecture that n is an integer iff it has an even number of divisors has been modified to: n is a non-square if and only if it has an even number of divisors.

4.2 Counterexample-barring

We ran two example sessions. In the first, we used an agency with two students and a teacher. The first student started with the integers 1-10 and the second student started with 11-20. Both students started with the background concepts of integers and divisors, and the forced concepts of prime numbers and odd numbers. The user set flags enabling both students to perform concept-barring and counterexample-barring. The teacher requested implication conjectures. The second student made and sent the conjecture that all primes are odd, which is true for the integers 11-20. The first student then found counterexample 2 and looked for a concept for which 2 was the only positive example. This failed, so the first student decided to use counterexample-barring. Hence it used the entity-disjunct rule to force the production of the concept of integers which are 2, and then used the negate production rule to force the concept of primes except 2. At the introduction of this concept, it made the conjecture that all primes except 2 are odd. Hence we see that the conjecture that all primes are odd has been modified to the conjecture that all primes except 2 are odd.

In the second example session, we again used an agency with two students and a teacher. The first student started with integers 1-10 and the second started with 11-20. Both students were given the background concepts of integers and divisors. In addition, we gave both students the forced concepts of even numbers and integers which are the sum of two primes. Again, the teacher requested implication conjectures. The second student made the conjecture that all even numbers are the sum of two primes, which is true of the integers 11-20. The first found the counterexample 2, and invented the concept of even numbers except 2 in response. This led it to the modified conjecture that all even numbers except 2 are the sum of two primes (Goldbach's conjecture).

4.3 Strategic Withdrawal

We ran the agency with two students and a teacher, where both students started with the integers 1-10 and the background concepts of integers, less than or equal, divisors, digit of, multiplication and addition. Also, each student was given the forced concepts of prime numbers

and odd numbers. The second student had the additional forced concept of odd non-squares. Both students were set to make near equivalences which held for 60% of the objects of interest. The user set flags enabling both agents to perform strategic withdrawal only, and the teacher requested near equivalence conjectures.

The first student made the near equivalence conjecture that: n is prime $\nleftrightarrow$ n is odd. The second student then found counterexamples 2, which is prime and not odd, and 1 and 9 which are odd but not prime. As it was set to perform strategic withdrawal, it then looked for a concept to cover the objects of interest which were positives for both the left hand and the right hand concepts, namely 3, 5, 7. It found that the concept of odd non-squares covered these examples precisely. Hence, it then made these modified conjectures:

$$n \text{ is odd } \wedge\, n \text{ is non-square} \rightarrow n \text{ is odd.}$$
$$n \text{ is odd } \wedge\, n \text{ is non-square } \rightarrow n \text{ is prime.}$$

Note that the first modified conjecture is an instance of a tautology and the second one is false – the first counterexample is 15. This is an illustrative example of strategic withdrawal in action, and it highlights that these methods are themselves heuristic: given the nature of the two modified conjectures produced, it seems that surrender would have been better here.

5 Conclusions and Further Work

We have given an account of automated theory formation and shown how it has been enhanced by ideas from Lakatos which has led to an implementation which (a) represents a social setting using a multi-agent system and (b) implements specific techniques for modifying a faulty conjecture in the light of known counterexamples. To the best of our knowledge, this is the first implementation of a system based on the ideas laid down by Lakatos. Moreover, the introduction of these methods represents a major step forward in our account of automated theory formation. To highlight this, we gave illustrated results of how the implemented techniques operate within the context of automated theory formation.

In terms of the two axes of development described in §2 and §3, namely the sophistication of the agent interaction and the sophistication of the conjecture-fixing techniques, we are more advanced along the latter axis. In particular, our agents cannot yet be thought of as autonomous, as the students do not yet make their own decisions about which Lakatos methods to employ. We are currently implementing such autonomous behaviour. With respect to the methods themselves, we have implemented techniques based on surrender, piecemeal exclusion and strategic withdrawal. In order to implement and test other Lakatos methods, it seems likely that we will have to concentrate on another domain. For instance, monster-barring is not particularly appropriate in number theory, as using this method would amount to stating that, for instance, the number 17 is not really an integer. Similarly, monster adjusting is not entirely appropriate in number theory.

It is our intention to pursue both axes of development to conclusion, and in doing so develop a reasoning system able to take advantage of all the techniques prescribed by Lakatos, carried out within a complex communication environment. We believe that the enhancements to theory behind and implementation of automated theory formation afforded by the methods described here will substantially increase the effectiveness of the system, and we hope to qualitatively demonstrate this soon.

As with the single version of HR, we propose to use a 'shotgun' approach to assess the value of our system. In particular, we will compare it with other approaches to fixing faulty conjectures, using example sets such as the one described in [15]. We will also attempt to objectively judge whether the system acts like the agency hypothesised by Lakatos, by giving detailed case studies. These will be performed in various domains in addition to the training domain – number theory. We will also perform extensive testing to determine how much the new system is an improvement on the single version of HR. That is, we will look at all aspects of theory formation and determine whether the new system is able to create a richer theory (i.e., search a more interesting/larger space) and/or create theories more efficiently.

We plan to improve the system in many ways. In particular, we hope to make the current techniques more flexible, for instance by enabling agents to look for concepts which cover some, but maybe not all of the examples. Also, we hope to enable agents to attempt to generate concepts which fit sets of objects of interest (counterexamples, etc.) rather than just looking for concepts existing in the theory already, i.e., using induction rather than abduction to identify covering concepts. This is essentially a machine learning problem, and we could employ machine learning techniques such as inductive logic programming (ILP) [16] to identify useful concepts. Note however, that HR itself has been fruitfully applied to such problems [7].

We believe that techniques for automated theory formation, and in particular the modification of hypotheses such as those presented here, are vital for the building of more intelligent reasoning systems. In particular, a current focus of attention in computer science is grid technology. The prospect of undertaking scientific endeavours with data, programs and resources distributed across the world is (purported to be) on the horizon. Even in our illustrative examples, we saw how the distribution of data between agents (i.e., the integers they were given) led to the introduction of faulty hypotheses. It is inconceivable to think that scientific projects using the grid will not run into similar inconsistencies, and robust systems for handling argumentation and cooperation between grid agents are being designed and built. Hence, it seems likely that something akin to the Lakatos-style methods we have implemented will be required for the projects to be carried out successfully.

Moreover, automated theory formation has much potential for the automatic reformulation of AI problem statements (pre-processing) and the reformulation of

problem answers (post-processing). In particular, HR has already been fruitfully used to discover new constraints for constraint satisfaction problems concerning quasigroups [9]. Moreover, we have recently begun experiments with HR pre-processing the input to the Progol ILP machine learning system, in such a way that useful features of the data are identified as potential targets to learn over. A potential use of the Lakatos methods is in post-processing machine learning outputs, which are essentially hypotheses. These hypotheses often have known counterexamples, and hence could be subject to re-formulation using the techniques described here.

However, perhaps the biggest potential for Lakatos-enhanced automated theory formation lies in the application to automated theorem proving. Of course, one reason for our optimism is that Lakatos's original notions were extracted from the (hypothesised) efforts of mathematicians attempting to prove a theorem. Also, as we stated earlier, we believe there needs to be more flexibility in the approach that theorem provers take. At present, in general it is expected that an ATP system will be given what is expected to be a theorem, with no noise in terms of (a) the concept definitions within the theorem statement (b) the structure of the theorem, and (c) the empirical evidence – if any – which was used to support the theorem in the first place.

We believe that – as with CSP solving and machine learning – ATP systems which are to be of use to domain scientists will need to handle noise, and be more flexible than they are at present. The Lakatos-enhanced system described here is designed to be such a flexible system. Although the emphasis is not on theorem proving itself, we believe that adding such Lakatos-enhanced theory formation abilities to a prover would substantially improve the flexibility and intelligence of the system. We envisage a sophisticated ATP system which can be given an ill-formed, possibly false conjecture – such as those that research mathematicians work with everyday – and through a collective reasoning process, produces a proved theorem that closely resembles the original. We believe that such functionality is essential to the future development of ATP and other reasoning systems.

Acknowledgements

This work has been supported by EPSRC grant GR/M45030. We would like to thank Alan Smaill and John Lee for their continued and much valued input. We are also grateful to the reviewers for their interesting comments and advice.

References

[1] A Bundy. *The Computer Modelling of Mathematical Reasoning.* Academic Press, 1983.

[2] D Burton. *Burton's History of Mathematics: An Introduction.* Wm. C. Brown, IA, USA, 1991.

[3] S Colton. Automated theory formation applied to mutagenesis data. In *Proceedings of the 1st British-Cuban Workshop on BioInformatics*, 2002.

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

[5] S Colton. The HR program for theorem generation. In *Proceedings of the Eighteenth Conference on Automated Deduction*, 2002.

[6] S Colton, A Bundy, and T Walsh. Agent based cooperative theory formation in pure mathematics. In *Proceedings of the AISB-00 Symposium on Creative & Cultural Aspects and Applications of AI & Cognitive Science*, 2000.

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

[8] 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.

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

[10] J Delgrande and J Mylopolous. Knowledge representation: Features of knowledge. In *Fundamentals of Artificial Intelligence: An Advanced Course, LNCS 232*, pages 3–36. Springer-Verlag, 1986.

[11] M Fisher and M Wooldridge. Distributed problem solving as concurrent theorem proving. In *Proceedings of the Eighth European Workshop on Modelling Autonomous Agents in a Multi-Agent World.* Springer-Verlag, 1997.

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

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

[14] 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.

[15] R Monroy, A Bundy, and A Ireland. Proof plans for the correction of false conjectures. In *Proceedings of the Fifth International Conference on Logic Programming and Automated Reasoning*, 1994.

[16] S Muggleton. Inverse entailment and Progol. *New Generation Computing*, 13:245–286, 1995.

[17] A Pease, S Colton, A Smaill, and J Lee. Semantic negotiation: Modelling ambiguity in dialogue. In *Proceedings of Edilog 2002, the 6th Workshop on the semantics and pragmatics of dialogue*, 2002.

[18] A Rao and M Georgeff. BDI agents: From theory to practice. Technical Report 56, Australian Artificial Intelligence Institute, Melbourne, Australia, 1995.