# Computational Discovery in Pure Mathematics

Simon Colton

Department of Computing
Imperial College
180 Queens Gate
London SW7 2BZ
United Kingdom
sgc@doc.ic.ac.uk
http://www.doc.ic.ac.uk/~sgc

**Abstract.** We discuss what constitutes knowledge in pure mathematics and how new advances are made and communicated. We describe the impact of computer algebra systems, automated theorem provers, programs designed to generate examples, mathematical databases, and theory formation programs on the body of knowledge in pure mathematics. We discuss to what extent the output from certain programs can be considered a discovery in pure mathematics. This enables us to assess the state of the art with respect to Newell and Simon's prediction that a computer would discover and prove an important mathematical theorem.

## 1 Introduction

In a seminal paper predicting future successes of artificial intelligence and operational research, Alan Newell and Herbert Simon suggested that:

> 'Within ten years a digital computer will discover and prove an important mathematical theorem.' (Simon & Newell, 1958)

As theorem proving involves the discovery of a proof, their predictions are about automated discovery in mathematics. In this chapter, we explore what constitutes knowledge in pure mathematics and therefore what constitutes a discovery. We look at how automated techniques fit into this picture: which computational processes have led to new knowledge and to what extent the computer can be said to have discovered that knowledge.

To address the state of the art in automated mathematical discovery, we first look at what constitutes mathematical knowledge, so that we can determine the

ways in which a discovery can add to this knowledge. We discuss these issues in Section 2. In Sections 3 to 7, we look at five broad areas where computational techniques have been used to facilitate mathematical discovery. In particular, we assess the contributions to mathematical knowledge from computer algebra systems, automated theorem provers, programs written to generate examples, mathematical databases, and programs designed to form mathematical theories.

We then return to Newell and Simon's prediction and consider whether important discoveries in mathematics have been made by computer yet. We conclude that no theorem accepted as important by the mathematical community has been both discovered and proved by a computer, but that there have been discoveries of important conjectures and proofs of well known results by computer. Furthermore, some systems have both discovered and proved results which could potentially be important. We further conclude that, with the ubiquitous use of computers by mathematicians and an increasing dependence on computer algebra systems, many important results in pure mathematics are facilitated – if not autonomously discovered – by computer. We then discuss possibilities for the use of other software packages in the mathematical discovery process.

### 1.1 Scope of the Survey

We restrict our investigation to pure mathematics, to avoid discussion of any application of the mathematics discovered. For instance, a program might invent an algorithm, which when implemented, leads to a better design for an aircraft wing. From our point of view, the algorithm is the discovery of importance, not the wing design. By restricting our investigation to pure mathematics, we hope to make this explicit.

(Valdés-Pérez, 1995) makes a distinction between (a) programs which have been designed to model discovery tasks in a human-like manner and (b) programs which act as scientific collaborators. While these two classes are certainly not mutually exclusive, there have been many mathematics programs which have not been used for discovery tasks. These include the AM and Eurisko programs (Lenat, 1982) (Lenat, 1983), the DC program (Morales, 1985), the GT program (Epstein, 1987), the ARE program (Shen, 1987), the Cyrano program (Haase, 1986), the IL program (Sims, 1990), and more recently the SCOT program (Pistori & Wainer, 1999) and the MCS program (Zhang, 1999). These systems are surveyed in (Colton, 2002b), but we restrict ourselves here to a survey of programs which have actually added to mathematics.

The Graffiti program, as discussed in Section 7.1, has been applied to chemistry (Fajtlowicz, 2001), and the HR program, discussed in Section 7.3, is currently being used in biology (Colton, 2002a). However, our final restriction in this survey is to look only at the mathematical applications of the discovery programs. For comprehensive surveys of automated discovery in science, see (Langley, 1998) and (Valdés-Pérez, 1999).

## 2 Mathematical Knowledge and Discoveries

We can classify knowledge in pure mathematics into ground knowledge about mathematical objects such as groups, graphs and integers, as well as meta-level knowledge about how mathematical explorations are undertaken. To a large extent, only the ground knowledge is communicated via the conferences, journals, and textbooks, with the meta-level knowledge discussed mainly between individuals and in a few books as listed in Section 2.2.

### 2.1 Ground Mathematical Knowledge

Many journal papers and textbooks in pure mathematics proceed with quartets of background information, concept definitions, theorem and proof. The background information usually states some results in the literature that provide a context for the new results presented in the paper.

Under the 'definition' heading, new concepts are defined and sometimes obvious properties of the concept and/or some examples satisfying the definition are provided. Concept definitions include, but are not limited to: classes of object, (e.g., prime numbers), functions acting on a set of objects to produce an output, (e.g., the calculation of the chromatic number of a connected graph (Gould, 1988)), and maps taking one set of objects to another, (e.g., isomorphisms in group theory (Humphreys, 1996)).

Following the concept definitions, a theorem is proposed as a statement relating known concepts and possibly some new concepts. Theorems include statements that one class of objects has a logically equivalent definition as another class, i.e., if-and-only-if theorems. For example: an even number is perfect – defined as being equal to twice the sum of its divisors – if and only if it is of the form $2^n(2^{n+1}-1)$ where $2^{n+1}-1$ is prime (Hardy & Wright, 1938). Theorems also include statements that one class of objects subsumes another, i.e., implication theorems. For example: all cyclic groups are Abelian (Humphreys, 1996). In addition, theorems include non-existence statements, i.e., that there can be no examples of objects with a given definition. For example: there are no solutions to the equation $a^n + b^n = c^n$ for positive integers $a, b$ and $c$ with $n > 2$ (Fermat's Last Theorem (Singh, 1997)).

Finally, a proof demonstrating the truth of the theorem statement completes the quartet. The proof is usually a sequence of logical inferences proceeding from the premises of the theorem statement to the conclusion (although other strategies exist, as discussed in Section 2.2). Any newly defined concepts which were not explicitly mentioned in the theorem statement will probably appear in the proof,

as the concepts may have been extracted from the proof in order to make it easier to understand.

In addition to concepts, theorems and proofs, authors sometimes present open conjectures which, like theorems, are statements about concepts, but which lack a proof, such as the open conjecture that there are no odd perfect numbers. These are provided in the hope that someone will one day provide a proof and turn the conjecture into a theorem. Algorithms are another type of ground mathematics. These provide a faster way to calculate examples of a concept than a naive method using the definition alone, e.g., the sieve of Eratosthenes for finding prime numbers. Finally, examples of concepts often appear in journal papers, particularly when examples are rare, or when finding the next in a sequence has historical interest, as with the largest prime number.

### 2.2 Meta-Mathematical Knowledge

Information about how mathematical explorations have been undertaken, and how to undertake them in general, is rarely found in published form. There are, of course, exceptions to this, and mathematicians such as Poincaré have written about how to do mathematics (as discussed in (Ghiselin, 1996)), with similar expositions from (Lakatos, 1976) and (Pólya, 1988).

There has also been research that makes explicit some ways to solve mathematical problems. Such problems are often posed as exercises in mathematics texts and usually involve either the solution of a fairly simple, but illustrative, theorem, or the construction of a particular example with some desired properties. For instance, Paul Zeitz suggests the 'plug and chug' method: if possible, make calculations which are relevant to the problem (e.g., put numbers into a formula), and examine the output in the hope of finding a pattern which leads to a Eureka step and the eventual solution (Zeitz, 1999). This approach is discussed in more detail in (Colton, 2000).

Some work has also been undertaken to make explicit certain strategies for proving theorems, and there are some known approaches such as proof by induction, proof by contradiction, and proof by reductio ad absurdum. In particular, the area of proof planning has made more formal the notion of a proof strategy and inductive theorem provers utilise these ideas (Bundy, 1988).

### 2.3 Discovery in Pure Mathematics

Before discussing computational approaches to discovery in pure mathematics, we must first discuss what constitutes a discovery. Naively, any algorithm, concept, conjecture, theorem or proof new to mathematics is a discovery. However,

we need to qualify this: a new result in mathematics must be important in some way. For instance, it is fairly easy to define a new concept in number theory. However, if there are no interesting provable properties of the concept and no obvious applications to previous results, the concept may be an invention, but it is unlikely to be accepted as a genuine discovery.

Similarly, any conjecture, theorem, proof or algorithm must be interesting in the context within which it is discovered. An important result may have application to the domain in which it was discovered; for example the definition of a concept may simplify a proof. Similarly, a conjecture, which if true, may demonstrate the truth of many other results. An important result may also be something without obvious application, but which expresses some unusual[1] connection between seemingly unrelated areas. The question of interestingness in mathematics is discussed more extensively in (Colton et al., 2000b) and (Colton, 2002b).

In addition to finding a new result, discoveries can be made about previous results. For instance, while the nature of pure mathematics makes it less likely to find errors than in other sciences, another type of discovery is the identification of an error of reasoning in a proof. For example, Heawood discovered a flaw in Kempe's 1879 proof of the four colour theorem,[2] which had been accepted for 11 years. A more recent example was the discovery that Andrew Wiles' original proof of Fermat's Last Theorem was flawed (but not, as it turned out, fatally flawed, as Wiles managed to fix the problem (Singh, 1997)).

Similarly, papers are often published with a shorter proof of a known result than any previously found. For instance, Appel and Haken's original proof of the four colour theorem (Appel & Haken, 1977) was criticised because it required a computer to verify around 1500 configurations, and as such was not repeatable by a human. Fortunately, Robertson et al. have discovered a much simplified proof (Robertson et al., 1996).

In predicting that a computer would discover a new theorem (which may involve new concepts, algorithms, etc.), Newell and Simon restricted their discussion to discoveries at the object level. It is true that most mathematical discoveries occur at the object level, but discoveries at the meta-level are certainly possible. In particular, a theorem may be important not because of the relationship it expresses among the concepts in the theorem statement, but rather because of an ingenious step in the proof that is applicable to many other problems. Similarly, a new generic way to form concepts – such as a geometric or algebraic construction – can also be viewed as a discovery. For instance, in the long term, Galois' thesis on the insolubility of polynomials was more important because he introduced the notion of groups – an algebraic construct which arises in a

[1] The mathematician John Conway is much quoted as saying that a good conjecture must be 'outrageous' (Fajtlowicz, 1999).

[2] This theorem states that every map needs only four colours to ensure that no two touching regions are coloured the same.

multitude of other domains – than the actual result, although that in itself was a major breakthrough, as (Stewart, 1989) explains.

To summarise, mathematical discoveries include three main activities:

- discovering an important (interesting/applicable) concept, open conjecture, theorem, proof or algorithm;
- revising a previous result, by for instance, the identification of a flaw or the simplification of a proof;
- deriving a new method, in particular a proof strategy or a construction technique for concepts.

### 2.4 Approaches to Computational Discovery

In the next five sections, we discuss some computational approaches that have led to discoveries in pure mathematics. Firstly, we deal with computer algebra packages, the most common programs employed in pure mathematics. Following this, we look at the various uses of automated theorem provers which have led to discoveries. The next category covers a broad range of general and ad-hoc techniques designed to find examples of concepts. In the fourth category, we examine how the use of mathematical databases can facilitate discovery. The final category covers a multitude of systems designed to invent concepts, make conjectures, and in general form a theory about a domain, rather than to pursue specific results. We look at the first two categories in terms of the types of problems solved. However, for the final three categories, because the techniques are more ad-hoc, we sub-categorise them in terms of the techniques themselves, rather than the problems the techniques are used to solve.

## 3 Computer Algebra Systems

Computer algebra systems are designed to perform complex mathematical calculations, including algebraic manipulations, polynomial solution, differentiation, and so on. Such systems include Maple (Abell & Braselton, 1994), Mathematica (Wolfram, 1999) and GAP (Gap, 2000). The systems are usually accompanied by large libraries of functions – many of which are written by the mathematicians who make use of them – which cover a wide range of domains.

Four common ways in which computer algebra systems are used to facilitate discovery in pure mathematics are:

- Verifying a result with examples before a proof is attempted;
- Providing examples from which a pattern can hopefully be induced in order to state a conjecture;
- Filling in the specifics of a theorem statement;
- Proving theorems.

The first three applications are so common that we can illustrate them with examples from our own experience. We illustrate the fourth application of computer algebra to discovery with an example from the work of Doron Zeilberger.

### 3.1 Giving Empirical Weight

Computer algebra packages can be very useful for adding empirical weight to a conjecture before attempting to prove the conjecture. That is, whenever a plausible conjecture arises and it is possible to generate counterexamples, some effort is usually expended trying to find such a counterexample before an attempt to prove the conjecture is made. For example, using techniques described in Section 7, we made the conjecture that perfect numbers are never refactorable (with a refactorable number being such that the number of divisors is itself a divisor). Using the fast routines for integers in the GAP computer algebra package, we verified this result for all perfect numbers from 1 to $10^{54}$. This gave us the confidence to attempt a proof, and we eventually proved this theorem (Colton, 1999). Of course, the alternate outcome is possible: a counterexample can be found, but we leave discussion of this until Section 5.

### 3.2 Presenting Data for Eureka Steps

Another way in which computer algebra systems can facilitate discovery is to produce examples from complex calculations and present the data in a way that lets the user notice patterns and possibly make a conjecture. To the best of our knowledge, there are no data-mining tools within computer algebra packages that could automate the pattern spotting part of the process. However, they are often equipped with visualisation packages, which can certainly help to highlight patterns in output. Furthermore, computer algebra systems include programmable languages, so the output can be tailored to enhance the chances of identifying a pattern.

A fairly trivial, but illustrative, example occurred when we used the Maple computer algebra system to help solve a problem posed by Paul Zeitz (Zeitz, 1999):

> Show that integers of the form $n(n+1)(n+2)(n+3)$ are never square numbers.

Following Zeitz's 'plug and chug' advice, we simply used Maple to calculate the value of the function $f(n) = n(n+1)(n+2)(n+3)$, for the numbers 1 to 4, giving:

$$f(1) = 24, f(2) = 120, f(3) = 360, f(4) = 840$$

As predicted by Zeitz, we noticed that the output was always one less than a square number, so proving this would solve the problem. We also used Maple in the proof, by guessing that $n(n+1)(n+2)(n+3)$ could be written as a quadratic squared minus one. Then we tried different quadratics until Maple agreed upon an equality between $(n^2+3n+1)^2-1$ and $n(n+1)(n+2)(n+3)$. We could, of course, have done all this entirely without Maple, but it illustrates how computer algebra systems can be used to find, and, in some cases, verify patterns in data. We discuss a different approach to Zeitz's problem in (Colton, 2000).

### 3.3 Specifying Details in Theorems

A third way in which computer algebra systems can aid discovery is by filling in details in theorem statements. For example, we became interested in *divisor graphs* of integers, which are constructed by taking an integer $n$ and making the divisors of $n$ the nodes of a graph. Then, we joined any two distinct nodes with an edge if one divided the other. Figure 1 show three examples of divisor graphs:

**Fig. 1.** Divisor graphs for the numbers 10, 12 and 14.

As both 1 and $n$ divide all the other divisors, it follows that, for every integer, the result is a connected graph. We became interested in the question: which integers produce planar divisor graphs? A planar graph can be drawn on a piece of paper in such a way that no edge crosses another, such as the first and third graphs in Figure 1. Kuratowski's theorem (Kuratowski, 1930) is used to determine whether or not a graph is planar.

To answer our question, we first realised that the divisor graph of an integer is dependent only on its prime signature: if an integer $n$ can be written as $n = p_1^{k_1} p_2^{k_2} \ldots p_m^{k_m}$, for primes $p_1 < p_2 < \ldots < p_m$, then the prime signature of $n$ is the list $[k_1, k_2, \ldots, k_m]$. Hence, as depicted in Figure 1, the numbers $10 = 2 \times 5$ and $14 = 2 \times 7$ have the same divisor graph because they have the same prime signature: $[1, 1]$. Furthermore, we determined that if an integer $n$ is divisible by

a non-planar integer, then $n$ itself will be non-planar, because its divisor graph will have a non-planar subgraph.

These two facts enabled us to approach our problem by looking at the divisor graphs of numbers with different prime signatures and seeing whether the divisor graphs are planar. As most prime signatures will produce non-planar divisor graphs, we reasoned that we could determine a boundary on those prime signatures producing planar divisor graphs. We wrote a small Maple program to construct the divisor graph for an integer, then used the built-in `isplanar` function to determine whether the graph was planar. Rather than thinking too hard about the exact strategy, we simply printed out 'planar' or 'non-planar' for the numbers 1 to 100, and looked at the occurrences of non-planar divisor graphs.

We found that the number 30 produced a non-planar divisor graph. The number 30 can be written as $pqr$ for three distinct primes $p, q$ and $r$. Hence, using the above reasoning about prime signatures and graphs with non-planar subgraphs being non-planar themselves, we concluded that all integers divisible by three or more primes must be non-planar. Hence we focussed on integers with 1 or 2 prime divisors and we easily identified a boundary for the planar divisor graphs: integers of the form $p^4$ produced non-planar divisor graphs, as did integers of the form $p^2q^2$.

Hence, we answered our question, and could state the theorem as follows:

> Only the number 1 and integers of the form: $p, p^2, p^3, pq$ and $p^2q$ for distinct primes $p$ and $q$ produce planar divisor graphs.

We see that, whereas the overall idea for the theorem was ours, using Maple let us easily fill in the specifics of the theorem statement. Note also that the proof of the theorem evolved alongside the actual statement. That is, there was no time when we stated a conjecture and tried to prove it. This method of discovery is common in mathematics, but publications are usually stated with the theorem and proof very much separate. Further details of this and related theorems are found in (Colton, 2002b), appendix C.

### 3.4 Proving Theorems

As highlighted by Doron Zeilberger's invited talk at the IJCAR-2001 conference, Levi Ben Gerson's treatise of 1321 had around 50 theorems with rigorous proofs proving what are now routine algebraic identities, such as: $(a + b)^2 = a^2 + 2ab + b^2$ (which took a page and a half to prove). Computer algebra systems can

now "prove" much more elaborate identities than these with simple rewriting techniques. Zeilberger argues that, if mathematicians are happy to embrace such theorems as routine enough for a computer to carry out the proof, then we should embrace all such computer generated proofs. That is, rather than being skeptical of theorems that require a computer to prove them (such as the four colour theorem, which we discuss in Section 5.5), he encourages such proofs, arguing that they are an order of magnitude more difficult than theorems that a mere human can prove (but still trivial, as they can be proved).

As an example, we can take Zeilberger's proof of Conway's Lost Cosmological Theorem (Ekhad & Zeilberger, 1997). This starts with the sequence of integers $1, 11, 21, 1211, 111221, \ldots$, which is obtained by describing in words the previous term, e.g., one, one one, two ones, one two (and) one one, etc. Remarkably, Conway proved that the length of the $(n+1)th$ term divided by the length of the $nth$ term tends to a constant $\lambda = 1.303577269\ldots$ known as *Conway's constant*. The generalised cosmological theorem states that starting with any non-trivial integer other than 22, and constructing the sequence in the same manner, will give the same ratio between lengths. Conway lamented that his proof of this theorem, and another by Guy, were lost (Conway, 1987).

Zeilberger chose not to prove this theorem with pen and paper, but rather to write a Maple package called HORTON (after John Horton Conway). The proof relies on the fact that most numbers can be split into halves that develop independently of each other as the sequence proceeds. There are a finite number of integers (atoms) that cannot be split in this way. By ranging over all possibilities for atoms, the HORTON program showed that all numbers eventually decompose into atoms after a certain number of steps, which proved the conjecture (Ekhad & Zeilberger, 1997). This proof is similar in nature to the proof of the four colour theorem described in Section 5.5, but was undertaken using a computer algebra systems, rather than by writing software specifically for the problem.

## 4 Automated Theorem Proving

One of the original goals of computer science and artificial intelligence was to write a program that could prove mathematics theorems automatically, and many systems have been implemented which can prove fairly complicated theorems. The ability to prove theorems is a powerful skill and automated theorem provers have performed a variety of discovery tasks in pure mathematics. These include proving established conjectures, improving proofs, finding new axiomatisations, and discovering new theorems. We concentrate here on discoveries made using deductive methods. We leave discussion of (counter)example construction methods – which have also solved theorems – until Section 5.

### 4.1 Proving Established Conjectures

The most extensive application of automated theorem proving to pure mathematics has been undertaken by the research team at Argonne laboratories, using various theorem provers, including EQP and, more recently, Otter (McCune, 1990).

Certainly the most famous theorem to be proved by an automated theorem prover is the Robbins Algebra conjecture, which McCune et al. solved using the EQP theorem prover (McCune, 1997). Herbert Robbins proposed that commutative, associative algebras are Boolean if they satisfy the extra condition that $n(n(x) + y) + n(n(x) + n(y)) = x$. These algebras became known as Robbins algebras and the question of whether they are Boolean defeated the attempts of mathematicians and logicians for more than 60 years.

In addition to providing the solution in 1996, automated reasoning techniques were also used during the development of this problem. In particular, on the advice of Wos, Winker used a combination of automated techniques and a standard mathematical approach to find two simpler conditions on the algebras, which, if true, would show that they are Boolean. The EQP program eventually proved that the second condition does in fact hold for Robbins algebras. The search for the proof took around eight days on an RS/6000 processor and used around 30 megabytes of memory.

EQP's successor, the Otter program (McCune, 1990), has also had much success discovering proofs to theorems in pure mathematics. In addition to finding new axiomatisations of algebras, as discussed in Section 4.3, Otter has been used to prove research theorems in algebraic geometry and cubic curves, lattice theory, Boolean algebras and quasigroup theory. A particularly fruitful partnership between McCune (Otter's author and principal user) and the mathematician Padmanabhan has developed. Padmanabhan has supplied many theorems relevant to his research that Otter has proved for the first time. Much of this work was written up in (McCune & Padmanabhan, 1996). A web page describing the discoveries due to the Argonne provers can be found at: `http://www-unix.mcs.anl.gov/AR/new_results/`.

### 4.2 Improving Proofs

One of the first applications of automated theorem proving was the use of Newell, Shaw and Simon's Logic Theory Machine (Newell et al., 1957) to prove theorems from Whitehead and Russell's Principia Mathematica. The program proved 38 of the 52 theorems they presented to it, and actually found a more elegant proof to theorem 2.85 than provided by Whitehead and Russell. (MacKenzie, 1995) points out that, on hearing of this, Russell wrote to Simon in November 1956:

> 'I am delighted to know that Principia Mathematica can now be done by machinery ... I am quite willing to believe that everything in deductive logic can be done by machinery.'

Newell, Shaw and Simon submitted an article about theorem 2.85, co-authored by the Logic Theory Machine, to the *Journal of Symbolic Logic*. However, it was refused publication as it was co-authored by a program.

More recently, Larry Wos has been using Otter to find smaller proofs of theorems than the current ones. To this end, he uses Otter to find more succinct methods than those originally proposed. This often results in detecting double negations and removing unnecessary lemmas, some of which were thought to be indispensable. (Wos, 1996) presents a methodology using a strategy known as resonance to search for elegant proofs with Otter. He gives examples from mathematics and logic, and also argues that this work also implications for other fields such as circuit design.

(Fleuriot & Paulson, 1998) have studied the geometric proofs in Newton's Principia and investigated ways to prove them automatically with the Isabelle interactive theorem prover (Paulson, 1994). To do this, they formalised the Principia in both Euclidean geometry and non-standard analysis. While working through one of the key results (proposition 11 of book 1, the Kepler problem) they discovered an anomaly in the reasoning. Newton was appealing to a cross-multiplication result which wasn't true for infinitesimals or infinite numbers. Isabelle could therefore not prove the result, but Fleuriot managed to derive an alternative proof of the theorem that the system found acceptable.

### 4.3 Finding Axiom Schemes

Another interesting project undertaken by the Argonne team aims to find different axioms systems for well known algebras, such as groups and loops (McCune, 1992) (McCune, 1993). In many cases, it has been possible to reduce the axioms to a single axiom. For instance, group theory can be expressed in terms of the multiplication and inverse operators in a single axiom:

$$\forall\ x, y, z, u \in G,\ (x(y(((zz')(uy)')x))') = u,$$

where $a'$ indicates the inverse of $a$. These results were achieved by using Otter to prove the equivalence of the standard group axioms with exhaustively generated formulae. Similar results have been found for different operators in group theory and for Abelian groups, odd exponent groups and inverse loops. (Kunen, 1992) has subsequently proved that there are no smaller single axioms schemes than those produced by Otter.

### 4.4 Discovering Theorems

With a powerful theorem prover, it is possible to speculate that certain statements are theorems, and discard those which the prover does not prove. As the prover is so efficient, an exhaustive search of theorems can be undertaken, without having to invent concepts or worry about notions of interestingness as is the case with the programs described in Section 7.

(Chou, 1985) presents improvements on Wu's method for proving theorems in plane geometry (Wu, 1984). In Chapter 4, he describes three approaches to using his prover to find new theorems: (i) ingenious guessing using geometric intuition (ii) numerical searching and (iii) a systematic approach based on the Ritt-Wu decomposition algorithm. Using the first approach, he found a construction based on Pappus' Theorem which led to a colinearity result believed to be new. Using the second method – suggesting additional points and lines within given geometric configurations – he started with a theorem of Gauss (that the midpoints of the three diagonals of a complete quadrilateral are collinear) and constructed a theorem about taking subsets of five lines from six, which he believed to be of importance. With the third method, he discovered a generalisation of Simson's theorem. (Bagai et al., 1993) provide a more general approach to automated exploration in plane geometry, although it appears that no new theorems resulted from this work.

## 5 Example Construction

The construction of examples of concepts has advanced pure mathematics in at least these three ways:

- by discovering counterexamples to open conjectures;
- by solving non-existence conjectures, either by finding a counterexample or by exhausting the search space to prove the conjecture;
- by finding larger (more complex) examples of certain objects, such as the largest primes.

We break down our overview of example construction according to the general techniques used to solve problems, rather than the type of problem solved.

### 5.1 Constraint Satisfaction Solving

Specifying a problem in terms of constraint satisfaction has emerged as a powerful, general purpose, technique (Tsang, 1993). To do this, the problem must be stated as a triple of: variables, domains for the variables, and constraints on the assignment of values from the domain to each variable. The solution of the problem comes with the assignment of a value (or a range of values) to each of the variables in such a way that none of the constraints are broken. There are various strategies for the assignment of variables, propagation of constraints, and backtracking in the search.

This approach has been applied to problems from pure mathematics, in particular quasigroup existence problems. For instance, the FINDER program (Slaney, 1992) has solved many quasigroup existence problems. (Slaney et al., 1995) used FINDER along with two different automated reasoning programs called DDPP (a Davis Putnam implementation, as discussed in Section 5.2) and MGTP to solve numerous quasigroup existence problems. For example, they found an idempotent type 3 quasigroup (such that $\forall\ a, b(a * b) * (b * a) = a$) of size 12, settling that existence problem. They had similar results for quasigroups of type 4 and solved many other existence questions, both by finding counterexamples and by exhausting the search to show that no quasigroups of given types and sizes exist.

Open quasigroup problems have also been solved with computational methods by a number of other researchers.[3] Constraint satisfaction techniques have been applied to other problems from combinatorial mathematics, such as Golomb rulers (Dewdney, 1985), and Ramsey numbers (Graham & Spencer, 1990). However, optimised, specialised algorithms usually out-perform the constraint satisfaction approach. For instance, while the constraint approach works well for Golomb rulers, all the actual discoveries have been made by specialised algorithms.

### 5.2 The Davis Putnam Method

This method is used for generating solutions to satisfiability problems (Davis & Putnam, 1960), (Yugami, 1995). It works by searching for an assignment of variables that satisfies all clauses in a formula expressed in conjunctive normal form. The procedure uses unit propagation to improve performance and works by choosing a variable in a clause containing a single literal, and assigning a value that satisfies the clause.

The MACE program (McCune, 2001) uses the Davis-Putnam method to generate models as counterexamples to false conjectures and has also been employed to

[3] With details at this web page:
`http://www.cs.york.ac.uk/~tw/csplib/combinatorial.html`.

solve some existence problems. As discussed in (McCune, 1994), MACE found a pair of orthogonal Mendelsohn triple systems of order 9. Also, MACE found a quasigroup of type 3 of order 16 with 8 holes of size 2. Furthermore, MACE has solved the existence problem of size 17 quasigroups of type 6 (such that $f(f(x,y),y) = f(x,f(x,y))$) by finding an example. Another Davis Putnam program which has been used to solve more than 100 open questions about quasigroups in design theory is SATO (Zhang et al., 1996), (Zhang & Hsiang, 1994). Also, as mentioned above, the DDPP program is an implementation of the Davis-Putnam method.

### 5.3 The PSLQ Algorithm

The PSLQ algorithm, as described in (Bailey, 1998), is able to efficiently suggest new mathematical identities of the form:

$$a_1x_1 + a_2x_2 + \ldots + a_nx_n = 0$$

by finding non-trivial coefficients $a_i$ if supplied with real numbers $x_1$ to $x_n$.

One application of the algorithm is to find whether a given real number, $\alpha$, is algebraic. To do this, the values $\alpha, \alpha^2, \ldots, \alpha^n$ are calculated to high precision and the PSLQ algorithm then searches for non trivial values $a_i$ such that

$$a_1\alpha + a_2\alpha^2 \ldots + a_n\alpha^n = 0$$

This functionality finds application in discovering Euler sums, which has led to a remarkable new formula for $\pi$:

$$\pi = \sum_{i=0}^{\infty} \frac{1}{16^i}\left(\frac{4}{8i+1} - \frac{2}{8i+4} - \frac{1}{8i+5} - \frac{1}{8i+6}\right).$$

Note that the formula was actually discovered by hand and the numbers found by computation. This formula is interesting as it can be used to calculate the $n$th hexadecimal digit of $\pi$ *without* calculating the first $n-1$ digits, as discussed in (Bailey et al., 1997). Until this discovery, it was assumed that finding the $n$th digit of $\pi$ was not significantly less expensive than finding the first $n-1$ digits. The new algorithm can calculate the millionth hexadecimal digit of $\pi$ in less than two minutes on a personal computer.

### 5.4 Distributed Discovery Projects

With the increase in internet usage in the last decade, many distributed attempts to find ever larger examples of certain concepts have been undertaken. In particular, the latest effort to find the largest prime number is the Great Internet Mersenne Prime Search (GIMPS), which is powered by parallel technology running free software available at `www.mersenne.org`. Since 1996, the GIMPS project has successfully found the four most recent record primes. The record stands with a prime number with over two million digits and there are predictions that GIMPS will find a billion-digit prime number by 2010.

The search for prime numbers has an interesting history from a computational point of view. In particular, in 1961 in a single session on an IBM7090, Hurwitz found two Mersenne primes which beat the previous record. However, due to the way in which the output was presented, Hurwitz read the largest first. There is now a debate as to whether the smaller of the two was ever the largest known prime: it had been discovered by the computer before the larger one, but the human was only aware of it after he had found a larger one. Opinions[4] are split as to whether a discovery only witnessed by a computer should have the honour of being listed historically as the largest prime number known at a given time.

While finding the next largest prime is not seen as the most important activity in pure mathematics, it is worth remembering that one of the first programs worked on by Alan Turing (actually written by Newman and improved by Turing (Ribenboim, 1995)) was to find large prime numbers. Moreover, finding primes may one day lead to the solution of an important number theory conjecture by Catalan: that 8 and 9 are the only consecutive powers of integers.

There are many similar distributed attempts, some of which have come to a successful conclusion. Examples include a search to find ten consecutive primes in arithmetic progression (`www.ltkz.demon.co.uk/ar2/10primes.htm`) and a distributed computation to find the quadrillionth bit of $\pi$ (which is a 0), as described at `http://www.cecm.sfu.ca/projects/pihex/pihex.html`.

### 5.5 Ad-hoc Construction Methods

Individual programs tailored by mathematicians to solve particular problems are ubiquitous. One famous instance, which can be considered under the umbrella of example generation, is the four colour theorem. As mentioned in Section 2.1 above, this theorem has a colourful history (Saaty & Kainen, 1986) and was eventually solved in 1976 with the use of a computer to check around 1500

[4] See `www.utm.edu/cgi-bin/caldwell/bubba/research/primes/cgi/discoverer/` for some opinions on the notion of discovery with respect to prime numbers.

configurations in an avoidable set (see (Saaty & Kainen, 1986) for details). To solve the conjecture, Appel and Haken used around 1200 hours of computing time to check the configurations, looking for (and not finding) a configuration which would break the four colour theorem. This was the first major theorem to be proved by a computer that could not be verified directly by a human. As discussed above, simplifications have since been made to the proof that has made it less controversial and the truth of the theorem is now generally accepted (Robertson et al., 1996).

Another set of ad-hoc computational methods were used to solve the existence problem of finite projective planes of order 10. Clement Lam eventually proved that there are none, but only with the help of many mathematicians who wrote numerous programs during the 1980s. The complexities of this problem and its eventual solution are beyond the scope of this survey, but full details are given in (Lam et al., 1989).

Finally, we mentioned in Section 3 that mathematicians use computer algebra programs to find counterexamples to conjectures they originally think to be true. As an example of this, in appendix C of (Colton, 2002b), we discuss the following conjecture:

> A refactorable number is such that the number of divisors is itself a divisor (Colton, 1999). Given a refactorable number, $n$, then define the following function: $f(n) = |\{(a, b) \in \mathbf{N} \times \mathbf{N} : ab = n \text{ and } a \neq b\}|$. Then $f(n)$ divides $n$ if and only if $n$ is a non-square.

We attempted to find a counterexample to this claim using the GAP computer algebra system, but found none between 1 and 1,000,000. After abortive attempts to actually prove the conjecture, we began to look for counterexamples again. We eventually found three counterexamples: 36360900, 79388100 and 155600676. Hence, in this case, we disproved a conjecture by finding a counterexample. Unless the theorem is of importance, these kinds of results are rarely published, but they still represent computer discoveries in mathematics.

## 6 Mathematical Databases

The representation of mathematical knowledge and its storage in large databases is a priority for many researchers. For example, the MBASE project (Kohlhase & Franke, 2000) aims to create a database of concepts, conjectures, theorems and proofs, to be of use to the automated reasoning community, among others. Simply accessing databases of mathematical information can lead to discoveries. Such events occur when the item(s) returned by a database search differ radically

from those expected. Because the data is mathematical, there is a chance that the object returned from the search is related in some mathematical way to the object you were actually looking for.

One particularly important database is the Online Encyclopedia of Integer Sequences,[5] which contains over 75,000 integer sequences, such as prime numbers and square numbers. They have been collected over 35 years by Neil Sloane, with contributions from hundreds of mathematicians. The Encyclopedia is very popular, receiving over 16,000 queries every day. The first terms of each sequence are stored, and the user queries the database by providing the first terms of a sequence they wish to know more about. In addition to the terms of the sequence, a definition is given and keywords are assigned, such as 'nice' (intrinsically interesting) and 'core' (fundamental to number theory or some other domain). Sloane has recorded some times when using the Encyclopedia has led to a conjecture being made. For instance, in (Sloane, 1998), he relates how a sequence that arose in connection with a quantization problem was linked via the Encyclopedia with a sequence that arose in the study of three-dimensional quasicrystals.

Another important database is the Inverse Symbolic Calculator,[6] which, given a decimal number, attempts to find a match to one in its database of over 50 million taken largely from mathematics and the physical sciences. Other databases include the GAP library of around 60 million groups, the Mizar library of formalised mathematics (Trybulec, 1989), the Mathworld online Encyclopedia,[7] and the MathSciNet citation and review server,[8] which contains reviews for more than 10,000 mathematics articles and references for over 100,000. (Colton, 2001b) provides a more detailed survey of mathematical databases

## 7 Automated Theory Formation

We collect here some ad-hoc techniques generally designed to *suggest* conjectures and theorems rather than prove them. This usually involves an amount of invention (concept formation), induction (conjecture making), and deduction (theorem proving), which taken together amount to forming a theory.

### 7.1 The Graffiti Program

The Graffiti program (Fajtlowicz, 1988) makes conjectures of a numerical nature in graph theory. Given a set of well known, interesting graph theory invariants,

[5] `http://www.research.att.com/~njas/sequences`
[6] `http://www.cecm.sfu.ca/projects/ISC/`
[7] `http://mathworld.wolfram.com`
[8] `http://www.ams.org/mathscinet`

such as the diameter, independence number, rank, and chromatic number, Graffiti uses a database of graphs to empirically check whether one sum of invariants is less than another sum of invariants. The empirical check is time consuming, so Graffiti employs two techniques, called the **beagle** and **dalmation** heuristics, to discard certain trivial or weak conjectures before the empirical test (as described in (Larson, 1999)). If a conjecture passes the empirical test and Fajtlowicz cannot prove it easily, it is recorded in (Fajtlowicz, 1999), and he forwards it to interested graph theorists.

As an example, conjecture 18 in (Fajtlowicz, 1999) states that, for any graph $G$:

$$cn(G) + r(G) \leq md(G) + fmd(G),$$

where $cn(G)$ is the chromatic number of $G$, $r(G)$ is the radius of $G$, $md(G)$ is the maximum degree of $G$ and $fmd(G)$ is the frequency of the maximum degree of $G$. This conjecture was passed to some graph theorists, one of whom found a counterexample. The conjectures are useful because calculating invariants is often computationally expensive and bounds on invariants may bring computation time down. Moreover, these types of conjecture are of substantial interest to graph theorists, because they are simply stated, yet often provide a significant challenge to resolve – the mark of an important theorem such as Fermat's Last.

In terms of adding to mathematics, Graffiti has been extremely successful. The conjectures it has produced have attracted the attention of scores of mathematicians, including many luminaries from the world of graph theory. There are over 60 graph theory papers published which investigate Graffiti's conjectures.[9]

### 7.2 The AutoGraphiX Program

(Caporossi & Hansen, 1999) have recently implemented an algorithm to find linear relations between variables in polynomial time. This has been embedded in AutoGraphiX (AGX), an interactive program used to find extremal graphs for graph invariants (Caporossi & Hansen, 1997). AGX has been employed to refute three conjectures of Graffiti and has also been applied to automatic conjecture making in graph theory. Given a set of graph theory invariants calculated for a database of graphs in AGX, the algorithm is used to find a basis of affine relations on those invariants. For example, AGX was provided with 15 invariants calculated for a special class of graphs called colour-constrained trees. The invariants included:

$$\alpha = \text{the stability number}$$
$$D = \text{the diameter}$$

[9] See `http://cms.dt.uh.edu/faculty/delavinae/research/wowref.htm` for a list of the papers involving Graffiti's conjectures.

$$m = \text{the number of edges}$$
$$n_1 = \text{the number of pending vertices}$$
$$r = \text{the radius}$$

The algorithm discovered a new linear relation between the invariants:

$$2\alpha - m - n_1 + 2r - D = 0,$$

which Caporossi and Hansen have proved for all colour-constrained trees (Caporossi & Hansen, 1999).

### 7.3 The HR Program

HR is a theory formation program designed to undertake discovery tasks in domains of pure mathematics such as group, graph and number theory (Colton, 2002b), (Colton et al., 1999). The system is given some objects of interest from a domain, such as graphs or integers or groups, and a small set of initial concepts, each supplied with a definition and examples. From this, HR constructs a theory by inventing new concepts using general production rules to build them from one (or two) old concepts. The new concepts are produced with correct examples and a predicate definition that describes some relation between the objects in the examples.

HR uses the examples of the invented concepts to make empirical conjectures about them. For instance, if it finds that the examples of a new concept are exactly the same those of an old one, it conjectures that the definitions of the two concepts are logically equivalent. In finite algebraic systems such as group theory, HR uses the Otter theorem prover (McCune, 1990) to prove the conjectures it makes. If Otter fails to prove a conjecture, HR invokes the MACE model generator (McCune, 1994) to attempt to find a counterexample. Any counterexamples found are incorporated into the theory, thus reducing the number of further false conjectures generated.

We have used HR in domains such as anti-associative algebras (with only one axiom – no triple of elements is associative). This made us aware of theorems which were new to us; for example, there must be two different elements on the diagonal of the multiplication tables and that anti-associative algebras cannot be quasigroups or have an identity (in fact, no element can have a local identity). More results from HR's application to discovery tasks are presented in Chapter 12 of (Colton, 2002b). More recently, as discussed in (Colton & Miguel, 2001), we have applied HR to the invention of additional constraints to improve efficiency when solving constraint satisfaction problems. For instance, HR conjectured and proved that Qg3-quasigroups are anti-Abelian; i.e., for each $a, b$ such that $a \neq b$, $a*b \neq b*a$. HR also discovered a symmetry on the diagonal of Qg3-quasigroups, namely that $\forall\ a, b\, (a * a = b \rightarrow b * b = a)$.

### 7.4 The NumbersWithNames Program

As discussed in (Colton, 1999) and (Colton et al., 2000a), one of the original applications of HR to mathematical discovery was the invention of integer sequences worthy of inclusion in the Encyclopedia of Integer Sequences. To be included, they must be shown to be interesting, so we also used HR to supply conjectures about the sequences it invented, some of which we proved. To augment the supply of conjectures, we enabled HR to search the Encyclopedia to find sequences which were empirically related to the ones it had invented. Such relationships include one conjecture being a subsequence (or supersequence) of another, and one sequence having no terms in common with another.

We have extracted this functionality into a program called NumbersWithNames, which HR accesses. NumbersWithNames contains a subset of 1000 sequences from the Encyclopedia, such as prime numbers, and square numbers, which are of sufficient importance to have been given names. The internet interface[10] lets the user choose one of the sequences or input a new one. The program then makes conjectures about the sequence by relating it to those in its database. It also attempts to make conjectures by forming related sequences and searching the Encyclopedia with respect to them. The new sequences are formed by combining the given sequence with 'core' sequences in the database. NumbersWithNames orders the conjectures it makes in terms of a plausibility measure which calculates the probability of the conjecture occurring if the sequence had been chosen at random along the number line.

Some examples of conjectures made using this approach are given in (Colton, 1999), (Colton et al., 2000a) and appendix C of (Colton, 2002b). An example is the theorem that if the sum of the divisors of an integer $n$ is a prime number, then the number of divisors of $n$ will also be a prime number. Another recent appealing example is that perfect numbers are pernicious. That is, if we write a perfect number in binary, there will be a prime number of 1s (the definition of pernicious numbers). More than this, the 1s will be first, followed by zeros (another relation found by the program). For example, the first three perfect numbers are 6, 28 and 496, and when written in binary, these are 110, 11100 and 111110000. This unobvious conjecture – which we proved – is typical of those found by NumbersWithNames.

## 8 Summary

We have described what constitutes knowledge in pure mathematics and surveyed some computational techniques which have led to new knowledge in this

[10] This is available at: `http://www.machine-creativity.com/programs/nwn`.

field. We do not claim to have covered all mathematical discoveries aided by computer, as there are hundreds or perhaps thousands of individual programs crafted to assist in the discovery process, many of which we never hear about. However, from the evidence presented, we can draw some general conclusions:

- Computational discovery occurs at the object-level. While proof planning and automated theorem proving in general help us understand proof strategies, no new ones suitable for mathematicians have resulted from this. Similarly, although some work in meta-level concept formation (Colton, 2001a) has been undertaken, and theory formation in general gives us a better understanding of the exploration process, no new concept formation or conjecture making techniques have come to light.
- Most discoveries identify new results rather than improving old results, although improving proofs is developing into a useful application of theorem proving.
- Most types of object-level mathematics have been discovered by computer, including concepts, examples of concepts, open conjectures, theorems and proofs. We have not given evidence of any algorithms being discovered by computer, but neither do we rule this out.
- While theory formation programs are designed to carry out most activities in mathematics, computer algebra systems appear to be the most equipped to do this in a way that might lead to genuine discovery. We have given evidence of computer algebra systems being used to generate examples, to prove and disprove conjectures, to fill in specifics in theorem statements, and to make new conjectures.
- Most major discoveries have been made via specialised programs, or by much manipulation of the problem statement into a form by which automated techniques were applicable. For example, while a general theorem prover solved Robbin's algebra, this was only after much human (and automated) theory formation about the problem.
- Most of the important discoveries have played on the ability of computers to perform massive calculations with high accuracy. However, the discoveries made by theory formation programs required relatively less computing power.

Given this summarization, we can return to Newell and Simon's prediction and suggest ways in which we can increase and improve the use of computers in the mathematical discovery process.

## 9 Discussion

To recap, Newell and Simon predicted that, by 1968, a computer would have discovered and proved an important theorem in mathematics. By 2002, however, it is difficult to claim that a computer has both discovered and proved a theorem which the mathematical community would describe as important. Certainly, the smallest axiomatisations of groups discussed, the new geometry theorems discussed in Section 4 and the results in quasigroup theory as discussed in Section 7 were both discovered and proved by computer. Furthermore, as it is difficult to assess discoveries at the time they are made (there may be hidden applications or implications), these results may be considered important one day.

Moreover, automated theorem provers have proved important theorems, in particular the Robbins Algebra theorem, but they did not discover them. Conversely, the NumbersWithNames program and Graffiti have discovered, but not proved theorems in number theory and graph theory, respectively, which have interested mathematicians. The theorems from Graffiti are particularly important, as they provide bounds for numerical invariants that increase efficiency in calculations.

In looking for possible reasons for the lack of success, we refer back to the title of Newell and Simon's paper in which they make their predictions: "Heuristic Problem Solving: The Next Advance in Operations Research". The key word here is, of course, *heuristic*. When asked about theorem provers, Simon (personal communication, 2002) lamented that the heuristic approach in theorem proving had dwindled, replaced largely by complete methods. While guaranteed to find proofs eventually, complete methods have difficulty in proving hard results because of combinatorial explosions in their searches.

Simon certainly had a point and his early work shows the undeniable utility of heuristics in artificial intelligence. Certainly, if automated theorem provers are to prove important results in pure mathematics, heuristic searches will play an important role. However, this criticism may be a little unfair on automated theorem provers, some of which do use heuristic techniques. More importantly, theorem provers have found greater application in hardware and software verification and safety critical systems – where it is imperative that the proof be formal and complete – than in pure mathematics.

This suggests another possible reason why there has been less success than we expected: a lack of interest from the mathematical community. It has become clear through personal communication with mathematicians and through organisation of workshops such as (Colton & Sorge, 2000), that many mathematicians believe that, for the moment, they have all the computer tools they want. Only a few pure mathematicians use automated theorem provers on a regular basis, including Padmanabhan, who has used Otter on many occasions. However, we

should note that Padmanabhan has never actually invoked Otter: he passes theorems to Otter's author, Bill McCune, who tunes settings, runs Otter, and emails proofs back. Without mathematicians using and improving automated tools, it seems unlikely that an important theorem will be discovered and proved in the near future.

Perhaps the major reason why Newell and Simon's prediction has not come true is the implicit assumption that the computer acts autonomously – somehow it finds an important conjecture and manages to prove it. We have repeatedly mentioned that many discoveries are made possible only through the use of software packages (in particular computer algebra systems), but that the computer only facilitates the discovery: it cannot be said to have made the discovery, as usually the result is known (at least sketchily) to the user before the computer is employed.

For autonomous discovery to occur, more research must be put into automated theory formation. Much more research has been undertaken into automated theorem proving than into concept formation and conjecture making. Hence, computers are more able to prove results suggested by their user than to intelligently suggest interesting new ones. While many theorem proving systems have been built by entire teams, the more general question of automated theory formation has only been addressed in an ad-hoc manner by individual researchers. As mentioned in Section 1.1, examples of AI research in this area include the AM, Eurisko, GT, ARE, IL, DC, SCOT, MCS and Cyrano programs. While some of these built on previous work (e.g., Cyrano was based on AM), they were all isolated projects, and with the exception of SCOT and MCS, they are no longer current.

## 10 Conclusions

We have supplied evidence that a number of computational techniques have led to discoveries in pure mathematics, and argued that computer algebra systems are the most widely used programs among research mathematicians, many of whom believe they have all the computational tools they require at the moment. Taken together, these insights suggest that we embed more discovery techniques into computer algebra systems, such as theorem proving in the Theorema system (Buchberger et al., 1997). Alternatively, we could enable computer algebra systems to utilise other computational processes via some network of systems, as in the MathWeb project (Franke et al., 1999) and the Calculemus project.[11] Certainly, to encourage more autonomous discoveries, we need more integrated systems, as in the HR project and the SCOTT theorem prover (Hodgson & Slaney, 2001), which combines the example generation power of FINDER with

[11] See `http://www.eurice.de/calculemus` for details of the Calculemus project.

the deductive power of Otter. Furthermore, if we embed general techniques such as resolution theorem proving, constraint solving, and the Davis Putnam procedure into computer algebra systems, this may enhance the mathematician's productivity, as they would have to spend less time hand crafting programs to complete a task.

The mathematicians Andrew Wiles and Doron Zeilberger represent two extremes of computer use in pure mathematics. Wiles famously proved Fermat's last theorem in 1995 (Singh, 1997), and was reported to have used a computer only to write up his results. Zeilberger, on the other hand, recommends that we teach students of mathematics to program rather than to prove and – drawing on Chaitin's results on the limits of mathematics (Chaitin, 1998) – argues that the mathematics we can actually prove is trivial. He states that we can only view non-trivial mathematics through computer experimentation, and we should rely on quasi-proof rather than formal deduction to determine whether or not to accept a theorem. Zeilberger regularly publishes papers co-authored with his computer, which he calls Shalosh B. Ekhad.

It is difficult to gauge the extent to which an average mathematician uses a computer. There are still mathematicians who do not use computers at all, and many who use them for email and/or typesetting only. However, computer algebra systems such as Maple and Mathematica are being employed increasingly by mathematicians for research and teaching purposes. Furthermore, due to the number of undergraduate and postgraduate computer algebra courses, it seems likely that every new university-trained mathematician will be literate in this topic. Moreover, computer algebra systems have, without doubt, enriched pure mathematics, and there are scores of theorems that would not have been stated or proved without computer algebra systems and other computational techniques.

With projects such as the Center for Experimental Mathematics at Simon Fraser University and journals like the *Journal of Experimental Mathematics*, computer enhanced discovery is now recognised as a worthy approach to pure mathematics. Newell and Simon's 1958 predictions were optimistic, but they were certainly not unachievable. Their prediction, for instance, that a computer would beat the world chess champion did eventually come true. We hope that, with a new generation of mathematicians and the increased power of the mathematical software available to them, Newell and Simon will soon be vindicated in their optimism about automated mathematical discovery.

## Acknowledgments

I would like to thank Derek Sleeman and the Departments of Computing Science and Mathematics at the University of Aberdeen for inviting me to talk about computational discovery in pure mathematics, which provided the backbone for

the work presented here. Ideas presented here have also arisen from conversations at the 2000 CADE workshop on the role of automated deduction in mathematics, the 2001 IJCAR workshop on future directions in automated deduction and the 2001 IJCAI Workshop on Knowledge Discovery from Distributed, Dynamic, Heterogeneous, Autonomous Sources. I would like to thank Alan Bundy and Toby Walsh for their continued input to this work and I am very grateful to Ursula Martin, Paul Cairns, Ian Gent, Geoff Sutcliffe, Larry Wos, and Doron Zeilberger for suggesting some mathematical discoveries made by computer. Finally, I would like to thank Herbert Simon for meeting with me and discussing many interesting topics, some of which have inspired this paper. This work was supported by EPSRC grant GR/M98012.

## References

Abell & Braselton, 1994. Abell, M., & Braselton, J. (1994). *Maple V handbook.* Academic Press.

Appel & Haken, 1977. Appel, K., & Haken, W. (1977). Every planar map is four colorable. *Illinois Journal of Mathematics*, *21*, 429–567.

Bagai et al., 1993. Bagai, R., Shanbhogue, V., Żytkow, J., & Chou, S. (1993). Automatic theorem generation in plane geometry. *Proceedings of the 7th International Symposium on Methodologies for Intelligent Systems, LNAI 689* (pp. 415–424). Springer-Verlag.

Bailey, 1998. Bailey, D. (1998). Finding new mathematical identities via numerical computations. *ACM SIGNUM*, *33(1)*, 17 – 22.

Bailey et al., 1997. Bailey, D., Borwein, M., Borwein, P., & Plouffe, S. (1997). The quest for pi. *Mathematical Intelligencer*, *19(1)*, 50–57.

Buchberger et al., 1997. Buchberger, B., Jebelean, T., Kriftner, F., Marin, M., Tomuta, E., & Vasaru, D. (1997). A survey of the theorema project. *International Symposium on Symbolic and Algebraic Computation* (pp. 384–391).

Bundy, 1988. Bundy, A. (1988). The use of explicit plans to guide inductive proofs. *Ninth Conference on Automated Deduction (CADE 9).* Springer-Verlag.

Caporossi & Hansen, 1997. Caporossi, G., & Hansen, P. (1997). *Variable neighbourhood search for extremal graphs. 1. The AutoGraphiX system.* (Technical Report Les Cahiers du GERAD G-97-41). École des Hautes Études Commerciales, Montréal.

Caporossi & Hansen, 1999. Caporossi, G., & Hansen, P. (1999). Finding relations in polynomial time. *Proceedings of the 16th International Joint Conference on Artificial Intelligence* (pp. 780–785).

Chaitin, 1998. Chaitin, G. (1998). *The limits of mathematics.* Springer-Verlag.

Chou, 1985. Chou, S. (1985). *Proving and discovering geometry theorems using Wu's method* (Technical Report 49). Computing Science, University of Austin at Texas.

Colton, 1999. Colton, S. (1999). Refactorable numbers - a machine invention. *Journal of Integer Sequences,* `www.research.att.com/~njas/sequences/JIS`, *2*.

Colton, 2000. Colton, S. (2000). Automated plugging and chugging. *Computation and Automated Reasoning* (pp. 247–248). A. K. Peters.

Colton, 2001a. Colton, S. (2001a). Experiments in meta-theory formation. *Proceedings of the AISB'01 Symposium on Creativity in Arts and Science.*

Colton, 2001b. Colton, S. (2001b). Mathematics: A new domain for data mining? *Proceedings of the IJCAI'01 Workshop on Knowledge Discovery from Distributed, Dynamic, Heterogeneous, Autonomous Sources*.

Colton, 2002a. Colton, S. (2002a). Automated theory formation applied to mutagenesis data. *1st British-Cuban Workshop on Bioinformatics*.

Colton, 2002b. Colton, S. (2002b). *Automated theory formation in pure mathematics*. Springer-Verlag.

Colton et al., 1999. Colton, S., Bundy, A., & Walsh, T. (1999). HR: Automatic concept formation in pure mathematics. *Proceedings of the 16th International Joint Conference on Artificial Intelligence* (pp. 786–791).

Colton et al., 2000a. Colton, S., Bundy, A., & Walsh, T. (2000a). Automatic invention of integer sequences. *Proceedings of the 17th National Conference on Artificial Intelligence* (pp. 558–563).

Colton et al., 2000b. Colton, S., Bundy, A., & Walsh, T. (2000b). On the notion of interestingness in automated mathematical discovery. *International Journal of Human Computer Studies*, *53(3)*, 351–375.

Colton & Miguel, 2001. Colton, S., & Miguel, I. (2001). Constraint generation via automated theory formation. *Proceedings of the 7th International Conference on the Principles and Practice of Constraint Programming* (pp. 572–576).

Colton & Sorge, 2000. Colton, S., & Sorge, V. (Eds.). (2000). *Cade'00 workshop on the role of automated deduction in mathematics*.

Conway, 1987. Conway, J. (1987). The weird and wonderful chemistry of audioactive decay. *Open Problems in Communication and Computation* (pp. 173–188). Springer-Verlag.

Davis & Putnam, 1960. Davis, M., & Putnam, H. (1960). A computing procedure for quantification theory. *Associated Computing Machinery*, *7*, 201–215.

Dewdney, 1985. Dewdney, A. (1985). Computer recreations. *Scientific American, December issue*, 16–26.

Ekhad & Zeilberger, 1997. Ekhad, S., & Zeilberger, Z. (1997). Proof of conway's lost cosmological theorem. *Electronic Announcements of the American Mathematical Society*, *3*, 78–82.

Epstein, 1987. Epstein, S. (1987). On the discovery of mathematical theorems. *Proceedings of the 10th International Joint Conference on Artificial Intellignce* (pp. 194–197).

Fajtlowicz, 1988. Fajtlowicz, S. (1988). On conjectures of Graffiti. *Discrete Mathematics 72*, *23*, 113–118.

Fajtlowicz, 1999. Fajtlowicz, S. (1999). The writing on the wall. Unpublished preprint, available from http://math.uh.edu/ clarson/.

Fajtlowicz, 2001. Fajtlowicz, S. (2001). Computer generated conjectures in mathematical chemistry. *Dimacs Working Group Meeting on Computer-Generated Conjectures from Graph Theoretic and Chemical Databases I*.

Fleuriot & Paulson, 1998. Fleuriot, J., & Paulson, L. (1998). A combination of nonstandard analysis and geometry theorem proving, with application to newton's principia. *Proceedings of the 15th International Conference on Automated Deduction, LNAI 1421*. Springer-Verlag.

Franke et al., 1999. Franke, A., Hess, S., Jung, C., Kohlhase, M., & Sorge, V. (1999). Agent-oriented integration of distributed mathematical services. *Journal of Universal Computer Science*, *5*, 156–187.

Gap, 2000. Gap (2000). *GAP reference manual*. The GAP Group, School of Mathematical and Computational Sciences, University of St. Andrews.

Ghiselin, 1996. Ghiselin, B. (Ed.). (1996). *The creative process*. University of California Press.

Gould, 1988. Gould, R. (1988). *Graph theory*. Benjamin Cummings.

Graham & Spencer, 1990. Graham, R., & Spencer, J. (1990). Ramsey theory. *Scientific American, July issue*, 112–117.

Haase, 1986. Haase, K. (1986). Discovery systems. *Proceedings of the 7th European Conference on Artificial Intelligence* (pp. 546–555).

Hardy & Wright, 1938. Hardy, G., & Wright, E. (1938). *The theory of numbers*. Oxford University Press.

Hodgson & Slaney, 2001. Hodgson, K., & Slaney, J. (2001). System description: SCOTT-5. *Proceedings of the 1st International Joint Conference on Automated Reasoning* (pp. 443–447).

Humphreys, 1996. Humphreys, J. (1996). *A course in group theory*. Oxford University Press.

Kohlhase & Franke, 2000. Kohlhase, M., & Franke, A. (2000). MBase: Representing knowledge and context for the integration of mathematical software systems. *Journal of Symbolic Computation*, *11*, 1–37.

Kunen, 1992. Kunen, K. (1992). Single axioms for groups. *Journal of Automated Reasoning*, *9(3)*, 291–308.

Kuratowski, 1930. Kuratowski, G. (1930). Sur la problème des courbes gauches en topologie. *Fund. Math*, *15-16*.

Lakatos, 1976. Lakatos, I. (1976). *Proofs and refutations: The logic of mathematical discovery*. Cambridge University Press.

Lam et al., 1989. Lam, C., Thiel, L., & Swiercz, S. (1989). The nonexistence of finite projective planes of order 10. *Canadian Journal of Mathematics*, *41*, 1117–1123.

Langley, 1998. Langley, P. (1998). The computer-aided discovery of scientific knowledge. *Proceedings of the 1st international conference on discovery science*.

Larson, 1999. Larson, C. (1999). Intelligent machinery and discovery in mathematics. Unpublished preprint, available from http://math.uh.edu/~clarson/.

Lenat, 1982. Lenat, D. (1982). AM: Discovery in mathematics as heuristic search. *Knowledge-Based Systems in Artificial Intelligence*. McGraw-Hill Advanced Computer Science Series.

Lenat, 1983. Lenat, D. (1983). Eurisko: A program which learns new heuristics and domain concepts. *Artificial Intelligence*, *21*, 61–98.

MacKenzie, 1995. MacKenzie, D. (1995). The automation of proof: A historical and sociological exploration. *IEEE Annals of the History of Computing*, *17(3)*, 7–29.

McCune, 1990. McCune, W. (1990). *The OTTER user's guide* (Technical Report ANL/90/9). Argonne National Laboratories.

McCune, 1992. McCune, W. (1992). Automated discovery of new axiomatizations of the left group and right group calculi. *Journal of Automated Reasoning*, *9(1)*, 1–24.

McCune, 1993. McCune, W. (1993). Single axioms for groups and abelian groups with various operations. *Journal of Automated Reasoning*, *10(1)*, 1–13.

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

McCune, 1997. McCune, W. (1997). Solution of the Robbins problem. *Journal of Automated Reasoning*, *19(3)*, 263–276.

McCune, 2001. McCune, W. (2001). *Mace 2.0 reference manual and guide* (Technical Report ANL/MCS-TM-249). Argonne National Laboratories.

McCune & Padmanabhan, 1996. McCune, W., & Padmanabhan, R. (1996). *Automated deduction in equational logic and cubic curves, lnai 1095*. Springer-Verlag.

Morales, 1985. Morales, E. (1985). DC: a system for the discovery of mathematical conjectures. Master's thesis, University of Edinburgh.

Newell et al., 1957. Newell, A., Shaw, J., & Simon, H. (1957). Empirical explorations of the logic theory machine: A case study in heuristic. *Proceedings of the Western Joint Computer Conference* (pp. 218–39).

Paulson, 1994. Paulson, L. (1994). *Isabelle: A generic theorem prover, lncs 828*. Springer-Verlag.

Pistori & Wainer, 1999. Pistori, H., & Wainer, J. (1999). Automatic theory formation in graph theory. *Argentine Symposium on Artificial Intelligence* (pp. 131 – 140).

Pólya, 1988. Pólya, G. (1988). *How to solve it*. Princeton University Press.

Ribenboim, 1995. Ribenboim, P. (1995). *The new book of prime number records, 3rd edition*. Springer-Verlag, New York.

Robertson et al., 1996. Robertson, N., Sanders, D., Seymour, P., & Thomas, R. (1996). A new proof of the four-color theorem. *Elecronic Resources of the American Mathematical Society*, *2*, 17–25.

Saaty & Kainen, 1986. Saaty, T., & Kainen, P. (1986). *The four-color problem: Assaults and conquest*. Dover.

Shen, 1987. Shen, W. (1987). *Functional transformations in AI discovery systems* (Technical Report CMU-CS-87-117). Computer Science Department, CMU.

Simon & Newell, 1958. Simon, H., & Newell, A. (1958). Heuristic problem solving: The next advance in operations research. *Operations Research*, *6(1)*, 1–10.

Sims, 1990. Sims, M. (1990). *IL: An Artificial Intelligence approach to theory formation in mathematics*. Doctoral dissertation, Rutgers University.

Singh, 1997. Singh, S. (1997). *Fermat's last theorem*. Fourth Estate.

Slaney, 1992. Slaney, J. (1992). *FINDER (finite domain enumerator): Notes and guide* (Technical Report TR-ARP-1/92). Australian National University Automated Reasoning Project.

Slaney et al., 1995. Slaney, J., Fujita, M., & Stickel, M. (1995). Automated reasoning and exhaustive search: Quasigroup existence problems. *Computers and Mathematics with Applications*, *29*, 115–132.

Sloane, 1998. Sloane, N. J. A. (1998). My favorite integer sequences. *Proceedings of the International Conference on Sequences and Applications*.

Stewart, 1989. Stewart, I. (1989). *Galois theory*. Chapman and Hall Mathematics.

Trybulec, 1989. Trybulec, A. (1989). Tarski grothendieck set theory. *Journal of Formalised Mathematics*, *0*.

Tsang, 1993. Tsang, E. (1993). *Foundations of constraint satisfaction*. Academic Press, London and San Diego.

Valdés-Pérez, 1995. Valdés-Pérez, R. (1995). Generic tasks of scientific discovery. *Working notes of the AAAI Spring Symposium on Systematic Methods of Scientific Discovery*.

Valdés-Pérez, 1999. Valdés-Pérez, R. (1999). Principles of human computer collaboration for knowledge discovery in science. *Artificial Intelligence*, *107(2)*, 335–346.

Wolfram, 1999. Wolfram, S. (1999). *The mathematica book, fourth edition*. Wolfram Media/Cambridge University Press.

Wos, 1996. Wos, L. (1996). *The automation of reasoning: An experimenter's notebook with OTTER tutorial*. Academic Press.

Wu, 1984. Wu, W. (1984). Basic principles of mechanical theorem proving in geometries. *Journal of System Sciences and Mathematical Sciences*, *4(3)*, 207–235.

Yugami, 1995. Yugami, N. (1995). Theoretical analysis of Davis-Putnam procedure and propositional satisfiability. *Proceedings of the 14th International Joint Conference on Artificial Intelligence* (pp. 282–288).

Zeitz, 1999. Zeitz, P. (1999). *The art and craft of problem solving*. John Wiley and Sons.

Zhang et al., 1996. Zhang, H., Bonacina, M., & Hsiang, H. (1996). PSATO: a distributed propositional prover and its application to quasigroup problems. *Journal of Symbolic Computation*, *21*, 543–560.

Zhang & Hsiang, 1994. Zhang, H., & Hsiang, J. (1994). Solving open quasigroup problems by propositional reasoning. *Proceedings of the International Computer Symposium, Hsinchu, Taiwan*.

Zhang, 1999. Zhang, J. (1999). MCS: Model-based conjecture searching. *Proceedings of the 16th Conference on Automated Deduction, LNAI 1632* (pp. 393–397). Springer-Verlag.