The NumbersWithNames Program

Simon Colton
Mathematical Reasoning Group
Division of Informatics
University of Edinburgh, UK
Email: simonco@dai.ed.ac.uk

Louise Dennis
School of Computer Science
and Information Technology
University of Nottingham, UK
Email: lad@cs.nott.ac.uk

Abstract

We present the NumbersWithNames program which performs data-mining on the Encyclopedia of Integer Sequences to find interesting conjectures in number theory. The program forms conjectures by finding empirical relationships between a sequence chosen by the user and those in the Encyclopedia. Furthermore, it transforms the chosen sequence into another set of sequences about which conjectures can also be formed. Finally, the program prunes and sorts the conjectures so that the most plausible ones are presented first. We describe here the many improvements to the previous Prolog implementation which have enabled us to provide NumbersWithNames as an online program. We also present some new results from using NumbersWithNames, including details of an automated proof plan of a conjecture NumbersWithNames helped to discover.

1 Introduction

The Encyclopedia of Integer Sequences[1] is one of the most useful and popular mathematics resources available on the internet. With the help of many mathematicians, Neil Sloane has collected over 60,000 sequences of integers along with information about them such as definitions, links, computer algebra code, etc. Because of the number and range of sequences in the Encyclopedia, there have been occasions when coincidences arising from its use have led to a connection between two different areas of mathematics (and other sciences) being made. For instance, in [Slo98], Sloane relates how a sequence which arose in connection with a quantization problem was linked via the Encyclopedia with a sequence arising from the study of three-dimensional quasi-crystals.

As part of the HR project [Col01], we have attempted to increase the possibility of such research conjectures being made. The HR program enables this by data-mining the Encyclopedia to find empirical relationships between sequences. This initial approach is detailed in [CBW00], where the emphasis was on producing conjectures about sequences which HR had also invented. For instance, HR invented the concept of integers for which the number of divisors is a prime number, and through data-mining, it also conjectured that numbers where the *sum* of divisors is a prime number have a prime number of divisors — a fact we were able to prove. Further results from this initial approach are given in [Col99].

[1] Available here: `http://www.research.att.com/~njas/sequences`

The previous Prolog implementation within HR was very basic. We have now changed the emphasis so that the user can choose the sequence about which to form conjectures from any in the Encyclopedia (or indeed, any they care to invent). We have also improved the way in which the program makes conjectures, and made it available as the 'NumbersWithNames' Java program which can be used online at: `http://www.machine-creativity.com/programs/nwn.`

Given a sequence S, chosen by the user, NumbersWithNames performs a four step process:

1. it identifies and invents sequences related to S
2. it makes conjectures about S and the related sequences
3. it prunes any uninteresting conjectures
4. it sorts the conjectures in order of decreasing plausibility

In §2, we detail how NumbersWithNames makes conjectures by finding relationships between the chosen sequence (and transformations of it) and those in the Encyclopedia. In §3, we detail a new measure of plausibility for these conjectures which has been generalised from two previous measures. This measure is used to both prune implausible conjectures and to sort those remaining so that the user can view the most plausible first. In §4, we present some new results from the program, and detail how we have used the $\lambda Clam$ proof planner [RSG98] to find a proof plan for a generalised conjecture suggested by results from NumbersWithNames.

2 Making Conjectures in NumbersWithNames

The Encyclopedia contains many different types of sequence. In particular, there are around 1000 number types, such as prime numbers, even numbers, odd numbers, etc. in the Encyclopedia which are sufficiently important to have been given a name in the mathematical literature, and NumbersWithNames works with these. This design consideration was for various reasons. First, all the sequences are downloaded as part of a Java archive file, so having 1,000 rather than 60,000 to download was preferable. Second, searching through 1,000 sequences repeatedly to find conjectures is possible in an acceptable time limit, but searching through 60,000 repeatedly is not. Third, and most importantly, conjectures about number types can be stated in a natural way, for instance: prime numbers are not multiples of four, or: odd refactorable numbers are square numbers (refactorable numbers are such that the number of divisors is itself a divisor [Col99]).

2.1 Finding Empirical Relationships

For a sequence S, chosen by the user, NumbersWithNames searches through the 1,000 number types, trying to find sequences, T, which are empirically related to S. The relationships it looks for are:

• **Subsequences,** i.e., all members of T that are within the range of S are actually in S. (The range of S is the part of the number line it occupies). For example, suppose S was the perfect numbers (equal to the sum of their proper divisors) and T was the even numbers. All the perfect numbers stored in the Encyclopedia which are in the range of the even numbers (in the Encyclopedia the range of the even numbers is 0 to 120) are themselves even. Hence NumbersWithNames would make the conjecture that all perfect numbers are even (a well known open conjecture).

- **Supersequences,** i.e., all members of S that are within the range of T are actually in T.
- **Disjoint sequences,** i.e., no member of S is a member of T. For example, none of the entries in the perfect numbers sequence are found in the odd number sequence, so the program conjectures that there are no odd perfect numbers.
- **Moonshine sequences,** i.e., there is a large integer (greater than 10,000) which is found in both S and T. NumbersWithNames notices if any large integer in S matches to within $\pm$ 2 with one in T. This is inspired by the 'monstrous moonshine' theorem relating group theory and elliptic modular functions [CN79]. This theorem — the proof of which gained Richard Borcherds a Fields Medal — was discovered when the numbers 196883 and 196884 were found in seemingly distinct areas of mathematics.

2.2 Transforming the Given Sequence

Often, there may be a very interesting conjecture about a sequence closely related to the sequence of interest. As a trivial example, the conjecture: "all prime numbers are odd" is not true. However, the conjecture: "all prime numbers except 2 are odd" is true. Hence, transforming the sequence into a set of closely related ones may produce more interesting conjectures. To find concepts related to a chosen sequence S, NumbersWithNames first looks for ones with similar names in the database. For instance, if S was prime numbers, it would find sequences such as Mersenne prime numbers, Mills prime numbers, additive prime numbers and so on, and make conjectures about those also.

Following this, NumbersWithNames invents concepts by both transforming S and by combining it with others. The transformations are limited at the moment to:

- **Adding one and taking one from the sequence,** e.g., the sequences of primes-plus-one: $3, 4, 6, 8, \dots$
- **Monster-barring,** e.g., the sequences of primes-except-2: $3, 5, 7, 11, \dots$
- **Finding difference sequences,** i.e., taking the differences between consecutive terms in the sequence.
- **Finding the binomial sequence,** i.e., taking the first term of the difference sequence, the first term of the difference-of-differences sequence and so on (known as the binomial transformation).

NumbersWithNames also combines S with those which have been assigned the keyword: "core" in the Encyclopedia of Integer Sequences[2]. It has two ways to combine a pair of sequences:

- **Conjunction,** e.g., combining the sequences of odd numbers and prime numbers into the sequence: "odd prime numbers".
- **Indexing,** e.g., combining the sequences of odd numbers and prime numbers by taking the prime numbers which have an index in the sequence which is an odd number, i.e., p_1, p_3, p_5, etc. where p_i is the i-th prime number.

The user decides the number of additional sequences the program introduces. They are given three options: the first invents no additional sequences, the second invents all additional sequences except those produced by combination with the core sequences, and the third invents all additional sequences.

[2] Sequences in the Encyclopedia all have associated keywords, including "core", "nice" and "hard".

3 Pruning and Sorting Conjectures

NumbersWithNames employs pruning techniques to reduce the number of uninteresting and trivial conjectures produced. Firstly, the program discards conjectures which follow from the definitions of the sequences involved. For instance, it throws away the conjecture that odd prime numbers are prime numbers, because, using the names of these sequences, it assumes that odd prime numbers are, by definition, a specialisation of prime numbers. Secondly, it discards a conjecture if it has already made a stronger conjecture. For instance, it throws away conjectures such as: "e-perfect numbers are refactorable numbers" if it has already made (or makes later) the conjecture: "e-perfect numbers are *even* refactorable numbers". This is because the first conjecture is subsumed by the second, stronger, conjecture. This functionality is similar to the 'echo' heuristic employed by the Graffiti program (see §5). Finally, the user is able to prune the conjectures further, by supplying text which must be (or must not be) in the definition/keywords of the sequences in the conjecture.

Even after pruning, the program often produces a plethora of conjectures. Therefore, we enabled it to present the most plausible ones first. The plausibility is calculated as the probability that the conjecture is not a coincidence. Given sequence S with terms $s_1, \ldots, s_k$ and sequence T, the plausibility of the conjecture that T is a subsequence of S is calculated as:

$$1 - \left(\frac{k}{s_k - s_1}\right)^X$$

where X is the number of terms of T in the range of S. For example, suppose S was the powers of two, which has these terms in the Encyclopedia:

$$1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384, 32768, 65536, 131072, 262144,$$
$$524288, 1048576, 2097152, 4194304, 8388608, 16777216, 33554432, 67108864, 134217728,$$
$$268435456, 536870912, 1073741824, 2147483648, 4294967296, 8589934592$$

Further suppose that NumbersWithNames conjectures that superperfect[3] numbers:

$$2, 4, 16, 64, 4096, 65536, 262144, 1073741824, 1152921504606846976,$$

are powers of two. There are 34 powers of two recorded in the Encyclopedia, ranging over the numbers 1 to 8589934592, so the probability of a number between 1 and 8589934592 being a power of two is $(34/8589934592) \approx 0.000000004$. We do not know whether 1152921504606846976 is a power of two or not, because the sequence of powers of two stops before it gets that far. However, the other 8 superperfect numbers certainly are powers of two, and the probability of this happening by coincidence is therefore 0.000000004^8, which is approximately 6×10^{-68}. Therefore, the conjecture that superperfect numbers are powers of two is extremely plausible, because the probability of it happening as a coincidence is very small indeed. NumbersWithNames calculates this plausibility measure for the supersequence and subsequence conjectures and a similar one for the disjoint conjectures. It then presents the conjectures in decreasing plausibility. This measure supersedes two previous measures, namely the 'term overlap' (number of

[3]Superperfect numbers are integers n such that $\sigma(\sigma(n)) = 2n$, with $\sigma(n)$ defined as the sum of the divisors of n.

terms shared by the sequence and sub-sequence) and 'range overlap' (proportion of the number line occupied by either sequence which is actually occupied by both), which are described in [CBW00]. We found that the plausibility measure, in addition to being easier to use than the two previous measures, also highlighted more interesting conjectures. For instance, if three small terms overlap in a sequence and subsequence, the conjecture scores 3 for term overlap. If however, three large terms overlap — a potentially more significant result — the conjecture still only scores 3 for term overlap. In contrast, the plausibility measure is low for the small overlapping terms and high for the large overlapping terms.

4 Results

We chose 10 sequences at random and used all the functionality of NumbersWithNames to form as many conjectures about each one as possible. The average time to complete the task using the Java Virtual Machine (JVM) in Internet Explorer version 5 on a 1000Mhz laptop, was around 90 seconds per sequence. As this is not an excessive time to wait, we have not investigated any more sophisticated algorithms for finding conjectures. There is, however, a problem with the JVM implemented in versions of Netscape, and we have experienced between 4 and 5 times slower execution with Netscape. This appears to be a problem that Sun Microsystems are aware of.

NumbersWithNames, while relatively new as an online program, is in fact the conclusion of a project which has been ongoing for a number of years, namely using automated techniques within the HR program to find conjectures about sequences in the Encyclopedia of Integer Sequences. There have been many interesting results from this analysis along the way. Some interesting theorems HR discovered (which we proved ourselves) include:

- If the sum of divisors of an integer is prime, then the number of divisors is prime.
- Refactorable numbers (as described in §2 above) are congruent to 0, 1, 2 or 4 mod 8.
- Every even[4] perfect number can be written in the form $lcm(a, \sigma(a))$ and in the form $\phi(b)(\sigma(b) - b)$ for some a and b [where $lcm(x, y)$ is the lowest common multiple of x and y, $\sigma(a)$ is the sum of divisors of a and $\phi(b)$ is the number of integers less than or equal to b which are co-prime to it].

Also, there are many new conjectures produced by NumbersWithNames which are awaiting investigation (i.e., a proof or disproof), for example:

- e-perfect numbers (where the sum of the exponential divisors of n equals 2n) are even refactorable numbers (where the number of divisors is itself a divisor).

Also, NumbersWithNames has made some moonshine conjectures, such as pointing out the unlikely coincidence that:

- 1073741823 is a Stirling number, 1073741824 is a superperfect number (and power of two), and 1073741825 is a Jacobsthal-Lucas number.

Despite encouragement by ourselves within the mathematical community, we not encountered a mathematician using NumbersWithNames as a research tool. We present three successful investigations below,

[4]Note that the conjecture as to whether there exists an odd perfect number is still open.

in the hope that such success will encourage researchers to use NumbersWithNames to supply conjectures about sequences they are interested in.

4.1 Pernicious Numbers

Jeremy Gow invented the notion of pernicious numbers, namely integers n where the number of 1s in the binary representation of n is a prime number. This continues in the tradition of odious numbers (odd number of 1s) and evil numbers (even number of 1s). We wished to find something of interest about these numbers, but with the previous implementation of the data-mining within HR, we only discovered that powers of two are *not* pernicious. This is trivially true, because powers of two in binary form are a one followed by zeros.

However, when we looked for conjectures about pernicious numbers with NumbersWithNames later, it produced 165 subsequence conjectures. We pruned these by keeping only the 'core' sequences conjectured to be a subsequence of pernicious numbers. This reduced the number to seven and only one of these was true (we found counterexamples to the others using the GAP computer algebra system [Gap00]). The true conjecture was very interesting, though:

Perfect numbers are pernicious.

Perfect numbers — equal to the sum of their proper divisors — are of great interest in number theory, and any result about their nature may be important. The reason this conjecture was not made previously by HR is because we always used a term overlap minimum of four or more for the conjectures. That is, we instructed HR to discard any subsequence conjectures where the two sequences shared fewer than four terms. The perfect numbers are: $6, 28, 496, 8128, \ldots$ and the largest pernicious number in the Encyclopedia is 100, hence the conjecture that perfect numbers are pernicious was discarded, because the empirical evidence for it amounted to only two terms: 6 and 28. In NumbersWithNames, however, this conjecture was given a plausibility of 38%. Hence, as only conjectures with 0% plausibility are discarded, the conjecture was observed in the output.

It was not obvious to us that perfect numbers — defined in terms of the sum of their divisors — should show any special characteristics when written in binary. On writing the perfect numbers in binary, we noticed the following pattern:

$$6 = 110, \quad 28 = 11100, \quad 496 = 111110000, \quad 8128 = 1111111000000$$

To cross-check, we later used NumbersWithNames to provide conjectures about perfect numbers, and it conjectured not only that perfect numbers are pernicious, but also that they are nialpdrome numbers of type 2 (such that, in binary, they are 1s followed by 0s). Hence, taken together, NumbersWithNames had made the conjecture that perfect numbers, when written in binary, comprise a prime number of 1s followed by zeros, and we see that in the examples above.

It turns out to be fairly easy to prove this theorem, given a result found in Hardy and Wright's standard number theory text [HW38], that even perfect numbers are of the form: $2^{n-1}(2^n - 1)$ where $2^n - 1$ is a prime (called a Mersenne prime). It is fairly easy to show that multiplying a number of the form 2^{n-1} with a number of the form $2^n - 1$ (with n the same in each), produces a number which, when written in binary, is n ones followed by $n-1$ zeros. On presenting this to an 'integer sequence fans' mailing list, the overall impression was that, while it was a pleasing result they had not seen before, because it followed easily from Hardy and Wright's theorem, it was just an example of how related the concepts in number theory are, and was unlikely to be of importance. This should not detract, however, from the fact that NumbersWithNames made us aware of this theorem, which added to the value of pernicious numbers, and that we are unlikely to have found it ourselves.

4.2 Zeitz Numbers

NumbersWithNames can provide insight which may help solve problems. As an illustrative example, we looked at a problem posed in [Zei99]:

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

Zeitz suggests 'plugging and chugging', i.e., putting numbers into the formula and seeing if the results suggest anything which may help solve the problem. We used NumbersWithNames to help with the discovery part. Putting $n = 1, 2, 3$ and 4 into the formula above resulted in: $24, 120, 360$ and 840. We then added this as a new sequence to NumbersWithNames (which it is possible to do online, without having to recompile the program), and called this number type: 'zeitz numbers'. Then we asked for conjectures about this sequence.

The first four conjectures, sorted in terms of plausibility, about zeitz numbers were:

1. zeitz numbers are highly composite numbers
2. zeitz numbers are super-abundant numbers
3. zeitz numbers are minimal(1) numbers
4. zeitz-plus-one numbers are square numbers

The first three conjectures did not help us solve the problem, but the fourth one states that adding one to zeitz numbers produces square numbers. This implies that zeitz numbers can never be square numbers, because square numbers are never 1 apart on the number line. Thus, if the conjecture made by NumbersWithNames is true, then the problem is solved. In [Zei99], Zeitz says that making this conjecture is the most important part of solving the problem, and the rest follows easily from this Eureka step, namely showing that $n(n+1)(n+2)(n+3)$ can be written as $(n^2 + 3n + 1)^2 - 1$.

4.3 Sqrt(n)-Rough Numbers

To encourage the 'integer sequence fans' to use NumbersWithNames, we have periodically used it to form some conjectures about sequences that were currently being discussed on that mailing list. On one occasion, discussion centred around a sequence invented by Knuth and Greene [GK90]: integers where the largest prime factor is less than the square root. For example, 8 is the first such number, because the largest prime factor is 2, which is less than $\sqrt{8}$. These are called sqrt(n)-rough numbers. NumbersWithNames made a series of conjectures that interested us, including:

- centred square numbers (of the form $2n(n+1)+1$) are sqrt(n)-rough-plus-one numbers

- hex numbers (of the form $3n(n+1)+1$) are sqrt(n)-rough-plus-one numbers
- star numbers (of the form $6n(n+1)+1$) are sqrt(n)-rough-plus-one numbers

All of these conjectures had plausibility 99% or 100%, and we note that if NumbersWithNames hadn't invented the sequence of sqrt(n)-rough-plus-one (by adding one to the original sequence), this series of conjectures would not have been brought to our attention. This highlights the need for the concept formation part of NumbersWithNames. We generalised this result to the following:

Given any two integers k and n such that $n^2 > k > 1$,
then the number $kn(n+1)$ will be a sqrt(n)-rough number.

We proved this result, and found that, unusually, we did not have to appeal to any results from number theory other than some simple facts about inequalities and square roots.

λ*Clam* [RSG98] is a higher-order proof planning system. It is a descendent of the *Clam* [BvHHS90] series, and is specialised for proof by induction, but is also intended to allow the rapid prototyping of automated theorem proving strategies. λ*Clam* works by using depth-first planning with *proof methods*. Each node in the search tree is a subgoal under consideration at that point[5]. The planner checks the preconditions for the available proof methods at each node and applies those whose preconditions succeed to create the child nodes. The plan produced is then a record of the sequence of method applications that lead to a trivial subgoal.

Proof methods are intended to act as partial tactic specifications for tactics in some object-level theorem prover. In *Clam* this was the *Oyster* constructive type-theory system, which was based on Nuprl. At present, λ*Clam* has no associated theorem prover. However the plans produced by λ*Clam* are at an equivalent level to "pen and paper" proofs produced by mathematicians. λ*Clam*'s proof methods are believed to be sound although they are not currently reducible to sequences of inference rule applications in some logic. Thus a λ*Clam* plan of the conjecture above would represent an equivalent guarantee of correctness to that provided by the hand proof already in existence. Proof method applications are governed by their preconditions (which may be either legal or heuristic in nature) and by a *proof strategy* which restricts the possible proof methods available depending on the progress through the proof. For instance, when involved in rewriting a goal using a selection of definitions and lemmas, we generally wish to attempt to rewrite as much as possible (i.e. simplify the goal as much as possible) by applying our rewriting method exhaustively before considering other procedures such as checking for tautologies.

λ*Clam* is, at present, ill-equipped to deal with problems which rely on a large body of previous results for their proof, but is better able to deal with problems where the proof follows primarily from the definitions of the concepts involved. As this was the case with the sqrt(n)-rough conjecture, we decided to investigate whether λ*Clam* could prove the result. Our aim was to show that λ*Clam* could be incorporated into the process to provide — for simple conjectures at least — this sort of proof automatically. Combination of systems such as λ*Clam* and NumbersWithNames are important both for the advancement of Artificial Intelligence and in order to attract mathematicians to use automated

[5]More accurately each node is the partial plan at that point but viewing this as the current subgoal is sufficient for this application.

tools, i.e., if researchers are supplied not with conjectures, but rather proved theorems, then this might encourage them to invest some time applying and even developing such automated techniques.

Proof methods and proof strategies are devised by observing common patterns in families of proofs. They are intended to represent generic mathematical processes applicable across a range of problems. λ*Clam* had not previously been applied to problems like the theorem about sqrt(n)-rough numbers shown above, and so we had to create a new proof strategy. The original hand proof of the result was used as a guide to the proof procedures involved but abstracted to a number of generic steps. These were perceived to be the use of rewriting with definitions and lemmas and the use of reasoning based on transitivity. Rewriting is a standard procedure, and methods supporting this were already available in λ*Clam*. Reasoning about transitivity had not, however, been tackled previously by the system.

We extended λ*Clam* with a simple proof method for reasoning about transitivity. The proof method's preconditions were as follows:

- We wish to prove $H \vdash A < B$ where H is a list of hypotheses and A and B are terms.
- $C < B'$ appears in the hypothesis list, H, or $C < B'$ is a known lemma and there exists a substitution σ on the free variables of B and B' such that $\sigma(B) = \sigma(B')$.

If these preconditions succeeded then the planner would add the subgoal $\sigma(H) \vdash \sigma(A) < \sigma(C)$ to the search tree. There is an equivalent case for replacing A by a new value. We prototyped a proof strategy which attempted to repeatedly apply symbolic evaluation (rewriting) and this transitivity method, backtracking where necessary. In order to make the search tractable, we had to modify the transitivity method so that it did not attempt to prove $H \vdash A < 1$ at any point. An expert λ*Clam* user was able to put together the transitivity method and the prototype proof strategy (interleave rewriting and transitivity reasoning exhaustively) in less than 2 days of work.

With this machinery, λ*Clam* automatically found a plan for the conjecture:

$$\forall k.\, \forall n.\, ((1 < \sqrt{k}) \wedge (k < n^2) \wedge (\sqrt{k} < n)) \rightarrow sr(k * (n * (n+1)))$$

(where $sr(x)$ means that x is sqrt(n)-rough). A number of standard lemmas about squares etc. were assumed to make this possible. Note that the condition $1 < \sqrt{k}$ explicitly rules out those cases where $k = 2$ or $k = 3$, where a different style of reasoning, based on substituting values into the theorem would have been required.

Ideally, we would like to test our proof strategy on a number of conjectures produced by NumbersWithNames, to see if it is sufficiently general to automatically plan a range of problems. We have not attempted this, as we believe the strategy to be limited in its abilities. However, we are currently developing techniques in λ*Clam* for the rapid combination of decision procedure techniques [JB01]. The proof for the theorem planned by λ*Clam* requires a combination of linear arithmetic with some rewriting. This is a major application of the decision procedure work we are involved in and we hope that a generic strategy for performing this task will be available in λ*Clam* shortly. We believe that, while our proof strategy provided a proof of concept that λ*Clam* could be used for conjectures from NumbersWithNames, it would be more robust in the long term to use a proof strategy based on our decision procedure work.

Clearly, the nature of proof strategy development requires a family of conjectures with proofs which may be examined for common patterns. There is work in the automated development of proof strategies [JKB00] which may, in future, allow proof planning systems to make a contribution even in new domains. The theorem proving work reported here is very preliminary in nature, but we feel that the speed (relative to other proof strategy developments) with which a proof strategy for a NumbersWithNames conjecture could be developed shows that $\lambda Clam$ could be usefully used when conjectures are being formed in well-understood domains.

5 Related Work

Compared to systems for proving theorems automatically, there have been relatively few programs designed to automatically make research conjectures of real interest to mathematicians. The Graffiti program [Faj88] has, to the best of our knowledge, been the only program which has successfully produced many conjectures of sufficient difficulty and importance to come to the attention of research mathematicians. Graffiti has been designed by mathematician Siemion Fajtlowicz to make conjectures of a numerical nature in graph theory. Given a set of well known graph theory invariants, 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. If a conjecture passes the empirical test and Fajtlowicz cannot prove it easily, he forwards it to interested graph theorists.

As an example, conjecture 18 produced by Graffiti stated that, for any graph G:

$$\begin{array}{c} \text{chromatic\_number}(G) \\ + \\ \text{radius}(G) \end{array} \leq \begin{array}{c} \text{maximum\_degree}(G) \\ + \\ \text{frequency\_of\_maximum\_degree}(G) \end{array}$$

This was passed to some graph theorists, one of whom found a counterexample. These types of conjecture are of substantial interest to graph theorists because they are easy to understand, yet they often provide a significant challenge to resolve. The conjectures are also useful because calculating invariants is often expensive and bounds on sums of invariants may help bring computation time down.

In terms of adding to mathematical knowledge, 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. Graffiti owes some of its success to the fact that the inequality conjectures it makes are of a difficult and important type, and that Fajtlowicz himself uses Graffiti and prunes and disseminates the results to many interested parties. In contrast to NumbersWithNames, to our knowledge, Graffiti is not available for mathematicians to experiment with themselves.

6 Conclusions and Future Work

We have described the NumbersWithNames program which produces interesting conjectures about sequences of integers in number theory. We have described the numerous advances over the version in HR, including a generalised plausibility measure, the ability to transform the given sequence into related ones to find conjectures about, and the ability to form moonshine conjectures. There are many more improvements we hope to make, including additional transformations of sequences, and enabling NumbersWithNames to interact with computer algebra systems to further empirically check the conjectures it makes using the computer algebra code supplied with some of the sequences in the Encyclopedia.

We have also reported new results from this data-mining approach. In contrast to Graffiti, however, where the main user is interested in the results, we are not number theorists, and hence, not only do we have less interest in the results, we are also not in a position to assess the implications, applications or importance of the conjectures NumbersWithNames produces. For this reason, we have made the program available to run online in the hope that research mathematicians and recreational mathematicians will use it. Hence, the next stage of the project is to attract mathematicians to work both with the program and with us. We have started this process by making conjectures about some sequences being discussed on the sequence fans mailing list, and hope to continue this approach by targeting various researchers with conjectures about sequences of particular interest to them. Finally, we have described how λ*Clam* has been used to plan a proof for a generalised conjecture which arose from a series of conjectures made by NumbersWithNames, and we hope to pursue this interaction. In particular, we intend to use the decision procedure techniques soon to be available.

In a seminal 1958 paper [SN58], Newell and Simon made the prediction that:

> 'Within ten years a digital computer will discover and prove an important mathematical theorem.'

In our opinion — while some important mathematical theorems have been proved by automated means, for example the Robbins algebra problem [McC97] — for various reasons this prediction has not yet come true. Furthermore, only through interaction between conjecture making programs such as NumbersWithNames, HR and Graffiti, and theorem provers/planners such as λ*Clam*, will Newell and Simon's prediction be fulfilled. We cannot even claim that NumbersWithNames and λ*Clam* have discovered and proved a theorem autonomously, not to mention an important theorem. However, this is a goal of our project, and one which we believe is within the grasp of modern computational techniques.

Acknowledgments

Simon Colton is also affiliated with the Department of Computer Science at the University of York, UK. We would like to thank the anonymous reviewers whose comments helped improve the final version of this paper. This work has been supported by EPSRC grants GR/M98012 and GR/M45030.

References

[BvHHS90] A Bundy, F van Harmelen, C Horn, and A Smaill. The Oyster-Clam system. In *Proceedings of CADE-10*, pages 647–648. Springer-Verlag, 1990.

[CBW00] S Colton, A Bundy, and T Walsh. Automatic invention of integer sequences. In *Proceedings of the 17th National Conference on Artificial Intelligence*, pages 558–563, 2000.

[CN79] J Conway and S Norton. Monstrous moonshine. *Bulletin of the London Mathematical Society*, 11:308 – 339, 1979.

[Col99] S Colton. Refactorable numbers - a machine invention. *Journal of Integer Sequences,* `http://www.research.att.com/~njas/sequences/JIS`, 2, 1999.

[Col01] S Colton. *Automated Theory Formation in Pure Mathematics.* PhD thesis, Division of Informatics, University of Edinburgh, 2001.

[Faj88] S Fajtlowicz. On conjectures of Graffiti. *Discrete Mathematics 72*, 23:113–118, 1988.

[Gap00] Gap. *GAP Reference Manual.* The GAP Group, School of Mathematical and Computational Sciences, University of St. Andrews, 2000.

[GK90] D Greene and D Knuth. *Mathematics for the Analysis of Algorithms.* Birkhäuser, 1990.

[HW38] G Hardy and E Wright. *The Theory of Numbers.* Oxford University Press, 1938.

[JB01] P Janičić and A Bundy. A general setting for combining and integrating decision procedures into theorem provers. *Journal of Automated Reasoning*, 2001. To appear.

[JKB00] M Jamnik, M Kerber, and C Benzmüller. Towards learning new methods in proof planning. In *Proceedings of the 2000 Calculemus Symposium: Systems for Integrated Computation and Deduction*, 2000.

[McC97] W McCune. Solution of the Robbins problem. *Journal of Automated Reasoning*, 19(3):263–276, 1997.

[RSG98] J Richardson, A Smaill, and I Green. System description: proof planning in higher-order logic with λCLAM. In *Proceedings of CADE-15*, pages 129–133. Springer-Verlag, 1998.

[Slo98] N J A Sloane. My favorite integer sequences. In *Proceedings of the International Conference on Sequences and Applications*, 1998.

[SN58] H Simon and A Newell. Heuristic problem solving: The next advance in operations research. *Operations Research*, 6(1):1–10, 1958.

[Zei99] P Zeitz. *The Art and Craft of Problem Solving.* John Wiley and Sons, 1999.