A Survey of Research in Automated Mathematical Conjecture-Making

C. E. Larson

Abstract. The first attempt at automating mathematical conjecture-making appeared in the late-1950s. It was not until the mid-1980s though that a program produced statements of interest to research mathematicians and actually contributed to the advancement of mathematics. A central and important idea underlying this program is the Principle of the Strongest Conjecture: make the strongest conjecture for which no counterexample is known. These two programs as well as other attempts to automate mathematical conjecture-making are surveyed—the success of a conjecture-making program, it is found, correlates strongly whether the program is designed to produce statements that are relevant to answering or advancing our mathematical questions.

1. Introduction

Attempts have been made to automate various abilities belonging to research mathematicians—abilities which include the invention of new mathematical concepts, as well as the conjecture and proof of new theorems.

What counts as a "conjecture" and, thus, success or failure of various programs that might be called "conjecture-making programs" is partly a terminological question. The word "conjecture" is used in various ways: a teacher might call a student's proposal for trisecting an angle with ruler-and-compass a "conjecture" even though he knows such a construction is impossible; a mathematician who proposes some non-novel proposition may be credited with having made a "conjecture," for instance if evidence suggests that he couldn't have known it had already been advanced previously; the term can also be used to apply unqualifiedly to certain of a mathematician's genuinely novel mathematical advances.

The first program to produce novel mathematical statements, a "conjecture-making program" in one sense of this term, was due to Hao Wang in the late 1950s. Wang was interested in programs that produced mathematical statements that were not only novel but were "interesting." What is wanted in scientific practice is a reasonably unambiguous criterion of when to count an object, event, &c., as having a certain property. (Effort in pursuit of this aim can have fruitful consequences—as evidenced by Einstein's criterion for when to count two events as occurring "simultaneously.") Wang's criterion was that the produced propositions had a

1991 *Mathematics Subject Classification.* Primary 68T99; Secondary 05C99.

Key words and phrases. Automated conjecture-making, artificial intelligence, graph theory.

certain semantic form (propositional schema in two schematic variables, neither of which represented a tautology). His Program II produced statements in the language of propositional logic. Upon judging a statement to be interesting his program only produced the statement upon determining it's truth. Wang viewed his work as an experiment which would ultimately be extended to more complex mathematical domains.

> The selection of interesting conjectures or theorems and useful definitions is less easily mechanizable. For example Program II gives only very crude results. It should be of interest to try to get better results along the same line. In more advanced domains, however, the the question seems to have a complexity of a different order.
>
> If we use a machine to grind out a large mass of proofs, then there seems be be some mechanical test as to the importance and centrality of concepts and theorems. If a same theorem or expression occurs frequently, then we may wish to consider the theorem interesting or introduce a definition for the expression.
> ...
>
> A more stable criterion may be this: A formula which is short but but can be proved only by long proofs is a "deep" theorem. [**57**, p. 259]

Of course, Wang's proposals here are for judging which *theorems* to count as important, central, or interesting. They do not provide any criterion for which statements, produced by some successor of his Program II, should be counted as conjectures and whose theorem-hood should be proved or disproved.

Other criteria for the "interestingness" of a proposition have also been proposed. One might define a statement to be interesting if it inspired mathematical research resulting in publication. Research leading to publication explained wholly for psychological or sociological reasons: It *could* be that a researcher investigates the truth of some given statement for the same reason that she might pursue a crossword puzzle, that is, for entertainment, for no scientific reason at all. She *could* also pursue the truth of this given statement wholly for the reason that it involved novel concepts and thus could result in many basic results and a dissertation. A reviewer may recommend publication simply because the publication under review is mathematically sound and written by a well-known, well-regarded mathematician. Ultimate publication is certainly a well-defined criterion—but whether it accurately captures what mathematicians mean when they say a mathematical statement is "interesting" is debatable.

What epitomizes mathematical research is that it contributes to the advancement of mathematics. The clearest advances are when our existing mathematical questions are answered. The determination of the truth-value of an "interesting" mathematical statement, however interesting-ness is defined, may or may not advance or answer any of our existing mathematical questions. Hence, the clearest way to define what to count as a "conjecture" is not in terms of concepts which may be extrinsic to our mathematical goals, but to define it intrinsically, directly in terms of our mathematical goals. For the purposes of this paper then, a (research) conjecture will be defined to be a new mathematical proposition the determination of whose truth would be relevant to the advancement of our existing mathematical

questions. (And, henceforth, the term "research conjecture" will be used when the need arises to unambiguously distinguish this use of the word "conjecture" from other uses of this word.) G. H. Hardy famously claimed that a mathematician's product should be judged by its "beauty" and "seriousness." He described the seriousness of a mathematical theorem

> in the *significance* of the mathematical ideas which it connects. We may say, roughly, that a mathematical idea is "significant" if it can be connected in a natural and illuminating way, with a large complex of other mathematical ideas. Thus a serious mathematical theorem, a theorem which connects significant ideas, is likely to lead to important advances in mathematics itself and even in other sciences. [**36**, p. 89]

Applied to novel mathematical statements, a conjecture, as here defined, would have some degree of "seriousness" (as Hardy meant the word).

The mere novelty of a statement cannot be a sufficient condition for conjecturehood—it is trivial to produce new statements (or to write a program to produce them). Being new though is a necessary condition. If a human or program today conjectured that there are infinitely many twin primes (pairs of primes of the form p and $p + 2$), it would not be a contribution to the advancement of mathematics as this conjecture is already known, discussed and researched—and we would not credit the human or program with having made the conjecture. A new formula for the exact number of primes up to n, a new formula for the exact number or even a bound for any mathematical quantity for which formulas are sought, a proposition that would imply the existence or non-existence of odd perfect numbers, would all count as conjectures on this definition. The definition of "conjecture" given here provides an unambiguous criterion, removed from the vagaries of our psychology and sociology, and explained immediately with reference to actual mathematical practice. Among statements which count as conjectures under this definition, there is a continuum of relevance: these conjectures are certainly not of equal value—knowledge of the truth-values of some of them will answer or advance more of our mathematical questions than others. (Similarly, with Hardy's criterion for the "seriousness" of a mathematical theorem, there is a continuum of seriousness—some theorems connect more mathematical ideas than others.)

The first program that did make research conjectures was Siemion Fajtlowicz's Grafitti which initially appeared in the mid-1980s. One major difference between this program and the preceding attempts to automate conjecture-making was that the statements produced by this program were, by design, likely to be bounds for invariants for which mathematicians seek and publish bounds. The most successful of these programs to date are those which are similarly likely to produce statements which address some existing mathematical problem. This principle is almost certainly central to the design of successful conjecture-making programs.

2. Wang's Program II and Lenat's AM

In 1959 Wang, better known for his early research in the automation of theorem-proving, was the first person to attempt to program a machine to make conjectures. Wang's main research interest was in automated theorem-proving: his best-known

achievement was a program that could prove all the theorems in the first five chapters of Russell and Whitehead's *Principia Mathematica*. But he also wrote a program, his Program II, to produce statements in the language of propositional logic. [**57**]

Wang began: "A natural question to ask is 'Even though the machine can prove theorems, can it select the theorems to be proved?' " [**57**, p. 249] He believed this question could be answered in the affirmative and then went on to explain his approach.

> A very crude experiment in this direction was made with some quite preliminary results. These will be reported here, not for their intrinsic interest but for suggesting further attempts on the same line. The motive is quite simple: by including suitable principles of triviality, the machine will only select and print out less trivial theorems. These may in turn suggest further principles of triviality; after a certain stage, one would either arrive at essentially the same theorems which have already been discovered and considered interesting, or find in addition a whole crowd of interesting new theorems. [**57**, p. 249]

Wang identified a central issue in the automation of conjecture-making: how the program should select statements to output among the possible statements in the language of some area of mathematics.

Wang first restricted the output of his program to statements essentially of the following two propositional schemata: "If P then Q," and "P and Q," where P and Q were placeholders for propositions involving a small, fixed number of propositional constants and logical connectives. Wang implemented two "principles of triviality." First, his program culled from the stream of statements those where either P or Q was a theorem. The program also culled those where P and Q were the same. The statements that passed these triviality tests were then checked for truth (which could be done in this case as there is a complete proof procedure for propositional logic). Only true statements were produced.

Wang reported that his program produced 14,000 nontrivial statements, 1,000 of which were actually theorems and stored. [**57**, p. 246] He published several dozen of these statements. One example[1] is:

$$[(P \wedge Q) \wedge R] \rightarrow [(P \rightarrow R) \leftrightarrow P].$$

His idea seemed to be that one criterion of interesting mathematical statements is that they are not trivially true. This was the lone criterion he implemented for culling the stream of candiadate conjectures: his program produced so many statements satisfying his criterion that he had to halt the operation of his program. [**57**, p. 250] Wang did not identify any of the produced statements to be actually "interesting", however he meant the word. It is hard to imagine any interesting, new statements in the language of propositional logic. Wang's work should be viewed as an experiment which was intended to be extended to "more advanced domains."

[1]This statement has been translated from the second listed statement on [**57**, p. 261] from the special language Wang implemented to a more familiar one.

While his Program II did not produce any (research) conjectures, Wang should be credited for his vision and being the first to identify and attempt to address a central problem in the field of automated mathematical conjecture-making.

The next programs which might be described as an attempt to automate mathematical conjecture-making is Douglas Lenat's AM and its successor Eurisko. [**43, 45, 46, 47, 48, 49, 47, 51, 50, 52**] Lenat first reported on this research in the mid-1970s.

AM and Eurisko have been both highly lauded and the subject of various controversies [**55, 56, 42**]. Lenat reported that his programs produced a number of mathematical statements, including such venerable conjectures and theorems as Goldbach's Conjecture and the Unique Factorization Theorem. The only point relevant here though is uncontroversial: Lenat's programs never made any (research) conjectures. None appear in his papers and he reported that "AM was not able to discover any 'new-to-mankind' mathematics purely on its own ...." [**43**, p. 6]

Since the earliest days of Artificial Intelligence research, there has been an (often contentious) split between researchers who are interested in the automation (by any means possible) of abilities normally taken to require intelligence, and researchers interested in building machines to "simulate" intelligent human behaviors (and thereby learn something about human psychology). This division appeared in the late-1950s among researchers in automated theorem proving. Newell, Shaw and Simon wrote a program, LT, which was able to prove 38 of the more than 200 propositional logic theorems of Russell and Whitehead's *Principia*. [**53**, p. 112] It was unable to prove the remaining theorems. LT was designed to simulate how humans prove these theorems: "[W]e wish to understand how a mathematician, for example, is able to prove a theorem even though he does not know when he starts how, or if, he is going to succeeed." [**53**, p. 109] Shortly thereafter, Wang wrote his Program I which proved all the aforementioned theorems as well as those of other logical forms and did so relatively quickly. [**57**] Wang's program exhibited a behavior that we ordinarily take to require intelligence—although simulating human intelligence was not his goal, and whether or not it did was not his concern.

Lenat's programs are best judged as attempts at simulating the production of mathematical conjectures. Lenat, like Newell, Shaw and Simon, was interested in the implications of his work for understanding human psycholgy. Lenat, in his dissertation describing AM, proposed several measures for evaluating the performance of this program. [**43**, pp. 124-125] These include strictly mathematical measures such as the program's original contributions to mathematical research as well as measures best described as psychological, including the implications of his program's performance for how to do math. Assessing the extra-mathematical value of his work is outside the scope of this paper. The value of his research on the automation of conjecture-making is minimal: his programs did not make any (research) conjectures and subsequent work has shown that the design features he considered necessary in conjecture-making programs (such as large numbers of heuristics) are not, in fact, necessary. Kenneth Hasse's Cyrano (an attempt to reimplement Lenat's AM) [**40**] and Susan Epstein's Graph Theorist (GT) [**17, 18, 19**] are also best described as aiming to "simulate" human conjecture-making. Neither of these programs produced any (research) conjectures. [**42**]

3. Fajtlowicz's Graffiti

Graffiti, a program conceived by Fajtlowicz at the University of Houston (and developed, from 1990 to 1993, with Ermelinda DeLaVina), was the first program to have actually made (research) conjectures. While the first paper on this program appeared more than fifteen years ago, the ideas underlying Fajtlowicz's work remain largely unknown. In particular, no attempts have been made to implement Graffiti's heuristics, whether in mathematical conjecture-making programs or in other domains where they could be of use. Fajtlowicz's program was first described in his series of papers, "On Conjectures of Graffiti."

The statements Graffiti has produced are largely novel. Since the 1980s, Fajtlowicz has maintained a list *Written on the Wall* (WoW) of hundreds of these statements (WoW was originally distributed to interested researchers by regular mail, and then email, but can now be found on the WWW[2]). Many of these statements are explicit bounds for quantities for which bounds are desired; others imply bounds. Mathematicians have published numerous papers on bounds for various graph-theoretic quantities (invariants), for instance, the independence number of a graph; Graffiti's statements have provided new bounds for many of these quantities and thus these statements are (research) conjectures.

While it is not relevant to the determination of whether or not Graffiti's statements count as conjectures (that is, that the statements it produced satisfy the criteria for conjecture-hood, as defined here), it is still worth noting that Graffiti's statements have inspired research by numerous mathematicians. There are numerous papers, theses, and dissertations in which these statements (or weaker or stronger variants) are proved or disproved.[3] Graffiti has proved to be a genuine contributor to the advance of mathematics: its collaborators include such well-known graph theorists as Noga Alon, Bela Bollobas, Fan Chung, Paul Erdös, Jerry Griggs, Daniel Kleitman, Laszlo Lovasz, Paul Seymour and Joel Spencer. [**3, 14, 20, 21, 34, 41**, among others]

Graffiti's first conjectures were in the field of graph theory. Its underlying ideas, as described in Fajtlowicz's papers [**23, 25, 27**, in particular], apply not just to graphs: Graffiti has also made conjectures in geometry, number theory, and chemistry—conjectures about the structure of fullerenes (as represented by their graphs) have already led to papers by, among others, the fullerene expert Patrick Fowler. [**33, 29, 30**] One of Graffiti's conjectures led to the discovery, by Fajtlowicz and this author, that the minimization of the independence number of a fullerene is the best known predictor of its stability. [**30**] Graffiti's machinery can be applied to make conjectures about any objects, mathematical or otherwise, that can be represented by a computer—that is, its fundamental ideas are domain independent.

The ideas underlying Graffiti's operation are spread across several of Fajtlowicz's papers and tersely described. The following description of the design and operation of this program is meant to collect these ideas and provide sufficient elaboration that interested researchers can implement them. This description has benefitted from numerous discussions with Fajtlowicz.

---

[2]WoW, as well as a list of conjectures about fullerenes, can be found on the WWW at: math.uh.edu/~clarson. A list of conjectures about benzenoids is available at Fajtlowicz's home page: math.uh.edu/~siemion

[3]A partial list can be found on the WWW at: cms.dt.uh.edu/faculty/delavinae/wowref.html.

Suppose conjectures about objects of a given type are desired, and that representations of some number of these objects (call them $O_1, O_2, \ldots, O_n$) are stored in the computer's memory. An invariant of these objects is a function which associates a number to each of the objects (and is necessarily independent of how the objects are represented to the machine). Let $\alpha_1, \alpha_2, \ldots, \alpha_r$ be computable invariants (for a given object O, $\alpha_i = \alpha_i(O)$). Let $f_1, f_2, \ldots, f_s$ represent operations of some algebraic system (these might include, for instance, the ordinary arithmetic operators "plus," "times," &c., or any other unary, binary or n-ary operators.) Any term, for instance, $f(\alpha_1, \alpha_2)$, represents a new numerical invariant. (If "plus" is an operation, then $\alpha_1 + \alpha_2$ is a term—representing a new invariant). Statements can then be formed from relations of these terms. If t and s are two such terms, the expression $t \leq s$—which should be interpreted as the statement, "For every object O (of the type of object under consideration), $t(O) \leq s(O)$"—is a candidate for a conjecture.

A computer can produce an endless stream of such expressions. The idea of Graffiti is to cull conjectures from this stream. This problem, discussed already by Poincaré[4] and first addressed by Wang, was redressed by Fajtlowicz—the initial version of Graffiti, like Wang's Program II, produced thousands of statements. Fajtlowicz wrote, regarding this first version of his program,

> The number of conjectures, particularly those that are completely trivial, is the main problem and more than half of the program consists of various heuristics whose purpose is deletion of trivial and otherwise noninteresting but true conjectures.[5] [**23**, p. 113]

Suppose conjectured upper bounds of the invariant t are desired, that is, conjectures of the form $t \leq s$. They are to be culled from the stream of relations $t \leq s_1$, $t \leq s_2$, $t \leq s_3$, &c. Two heuristics are typically used in this task.

Dalmatian is Fajtlowicz's name for Graffiti's main heuristic for culling the stream of candidate conjectures. [**27**, pp. 370-371] Given a statement of the form $t \leq s$ and a (possibly empty) database of pre-existing conjectures of similar form, $t \leq u_1$, $t \leq u_2$, ..., $t \leq u_l$—the Dalmatian heuristic checks if the statement "$s(O) < u_1(O)$ and $s(O) < u_2(O)$ and ... and $s(O) < u_l(O)$" is true for at least one of the objects O from the set $O_1, \ldots, O_n$. That is, it checks if there is an object for which the value of the candidate upper bound is less than the minimum value of the existing conjectured upper bounds. If it is, then, with respect to the objects stored in memory, the relation $t \leq s$ says something informative—that is, the

---

[4]Hadamard writes:

> [I]t is obvious that invention or discovery, be it in mathematics or anywhere else, takes place by combining ideas. Now, there is an extremely great number of such combinations, most of which are devoid of interest, while, on the contrary, very few of them can be fruitful. Which ones does our mind—I mean our conscious mind—perceive? Only the fruitful ones, or exceptionally, some which could be fruitful.
> However, to find these, it has been necessary to construct the very numerous possible combinations, among which the useful ones are to be found. [**35**, pp. 29-30]

Hadamard attributes these ideas to Poincaré.

[5]Fajtlowicz points out that, with the addition of the Dalmation heuristic, this problem has been solved.

relation says something that was not implied by the totality of the previous conjectures of that form that had been kept in the program's database—so the relation remains a candidate for Graffiti to add to the database of conjectures. Otherwise, Graffiti rejects the relation as a possible conjecture—with respect to the databases of objects and pre-existing conjectures it is uninformative.

The second heuristic, applied to those relations which survive the Dalmatian heuristic, is to test for the truth of the relation with respect to the stored objects. If the relation is true of all of these objects then it is added to the database of conjectures; and if the relation is false for any of these objects then the general statement that the relation of term functions (the relation of invariants) represents is false—and the relation is not accepted as a conjecture. These first two heuristics are the heart of the program and express the following principle of Fajtlowicz: make the strongest conjecture for which no counterexample is known.

There are two other heuristics which Graffiti can also employ. One is applicable only when objects of a proper superclass of objects are already stored in the computers memory, the *Echo* heuristic. [**24**, p. 190] Suppose the database of objects includes $O_1, \ldots, O_m$, $O_{m+1}, \ldots, O_n$ of a type A and conjectures are desired of a type B, a subclass of A. Suppose the objects of type B are the objects $O_1, \ldots, O_m$. The Echo heuristic is used to cull those possible conjectures which are true of each of the objects $O_{m+1}, \ldots, O_n$: conjectures which could be true of all objects of type A—when what is desired are conjectures about its proper subclass B—are not specific enough and are rejected.

Graffiti's *Beagle* heuristic was central to early versions of the program. [**25**, pp. 23–24] Its function was largely superseded with the introduction of the Dalmatian heuristic. The Beagle heuristic was designed to avoid the endless numbers of conjectures like $\alpha \leq \alpha + 1$ where the relation is true but uninformative. The Beagle heuristic accomplished this by rejecting relations where the related terms (in our example, α and $\alpha + 1$) are very "close" to each other in the tree representing all possible terms: technically, the possible terms form a rooted tree and a distance can be defined on this tree—if the distance between two terms is too small, the Beagle heuristic rejects their relation as a possible conjecture. The Dalmatian heuristic rejects some but not all of these relations: it rejects those relations that are uninformative with respect to the existing conjectures and database of objects, but it makes allowances for certain relations the Beagle heuristic would have rejected—it will accept those relations that are informative, regardless of the closeness of its terms.

Graffiti's conjectures may, naturally, be false. These can be removed by informing the program of a counterexample, that is, by adding a new representation to the program's database of objects. Counterexamples can be found automatically, by producing representations of objects of the given type and testing the stored conjectures against them, or counterexamples can be provided by another intelligent agent—whether human or another computer.

Graffiti's operation is sped along by keeping its databases of objects and conjectures relatively small. The program only stores (representations of) objects which it has found "informative," that is, which have served as a counterexample to some previously made conjecture. Graffiti's database of conjectures is kept relatively small by eliminating conjectures that are no longer informative. Whenever a new relation is added to the database of conjectures, it is possible that one or more

pre-existing relations are no longer informative (with respect to the objects stored in the computer): if there is an object for which a bound is better than those given by all the other conjectured bounds then this bound is kept, otherwise it is removed (as, with respect to the stored objects, it does not improve on the other existing bounds). Note that this implies that the number of conjectured bounds of a given form (for instance, upper bounds for a given invariant), stored at a given moment, can never exceed the number of objects stored at that moment.

The removed bounds may be moved to a secondary database. If a counterexample is later found for one of the relations in the primary database of conjectures, then those relations in the secondary database are the first to be reconsidered as candidates for conjectures (that is, as candidates for the primary database) rather than arbitrarily formed relations. Graffiti's may been seen as mimicking the brain's operation in the sense that our brains do not and cannot store records of all the specific objects and relations holding between those objects that we encounter and experience; only some of those objects and relations make an "impression." Furthermore, another thing that we do is fall back on previously accepted but superseded beliefs when our present beliefs are proved wrong.

Graffiti has to date been used primarily to generate conjectures about *graphs* (a graph is a set of *vertices* together with a set of *edges* and an incidence function mapping edges to pairs of vertices). In this case the *objects* are graphs—one way to represent these in a computer is with certain matrices of 0's and 1's (the adjacency matrix of the graph). Numerical invariants for graphs include the *independence number* of the graph—the cardinality of the largest set of vertices such that no two are incident with the same edge. This invariant is of great practical and theoretical importance. The independence number of a graph cannot be computed efficiently (it is NP-hard). Researchers use existing upper and lower bounds for both theoretical and practical purposes. They can be used, for instance, to speed up computation of this invariant (one way is by using them to prune the recursion tree of the well-known recursive algorithm for computing the independence number).

Let α represent the independence number of a graph. Then conjectures regarding upper bounds of this graph invariant would have the form $\alpha \leq t$ (where t is a term function representing some other invariant), and conjectures regarding lower bounds would have the form $t \leq \alpha$. One of the first conjectures Graffiti made (WoW-2) was

$$\overline{d} \leq \alpha,$$

where $\overline{d}$ represents the *average distance* between distinct vertices of the graph. (This conjecture was proved by Fan Chung [**14**] in one of the first papers ever published about a computer-generated conjecture.) The conjecture was made using a database of connected graphs (for average distance is not defined for graphs in general) and should be interpreted as follows: "For every connected graph, the average distance between any two different vertices is no more than its independence number."

Among other things, Grafitti's success demonstrates that conjecture-making programs neither require huge numbers of heuristics nor domain-specific heuristics—which disproves Lenat's contention that programs which can make conjectures in multiple domains require domain-specific heuristics. [**45**, p. 288] [**49**, p. 223] No version of Graffiti has employed more than the four aforementioned heuristics—while making conjectures in graph theory, number theory, and chemistry.

It is worth noting here that Graffiti-like programs can be used to allow students to investigate subjects "Texas-style" (that is, using the teaching method made famous by R. L. Moore at the University of Texas). Fajtlowicz has experimented with using his program as part of a Texas-style approach to teaching graph theory. [**28**, pp. 22–25][**54**] and with a version of his program that his students can use to generate conjectures on their own. Students are able to explore Graph Theory by using the program to produce conjectures; the students then either attempt to prove or refute the conjectures. Upon refuting a conjecture and informing the program of a counterexample, the program then produces updated conjectures in response. DeLaVina has written a partial reimplemention of Graffiti on the Windows platform, Graffiti.pc [**16**], and has also experimented with using it as a teaching tool. [**16**, pp. 28–29]

4. Colton's HR and Caporossi and Hansen's AGX

Three other programs have been produced which have attempted to automate mathematical conjecture-making. One is the unnamed program of Bagai and his Wichita State colleagues [**2**]. They reported that their program "rediscovered" some known theorems in geometry—but did not report that the program produced any conjectures (which, as defined here, are necessarily new propositions).

In the late 1990s Simon Colton and his colleagues at the University of Edinburgh developed the program HR which, among other things, has produced new mathematical propositions. [**5, 6, 15**] Colton's team has attempted to automate all aspects of mathematical research—including the automation of conjecture-making.

The papers describing HR contain roughly a dozen examples of the mathematical propositions that the program has produced. The following three propositions are representative of the reported output of the program and also represent the three types of new propositions HR has produced.

1. For groups G and G' up to order 6, G and G' are isomorphic if and only if $f(G) = f(G')$, where the function f is defined as follows: for any group G,
$$f(G) = |\{(a, b, c) \in G^3 : a * b = c \textit{ and } b * c = a\}|.$$ [**6**]
2. No perfect number is refactorable[6]. [**15**]
3. The refactorable numbers are a subsequence of the sequence of positive integers congruent to 0, 1, 2, or 4 (mod 8). [**15**]

These statements are novel and thus satisfy one criterion for conjecture-hood. Would knowledge of the truth of these statements would advance any of our mathematical questions, that is, are they (research) conjectures? While it is possible, HR's authors do not claim that they do. HR can be directed to search for a structural characterization of some class of objects. This is the genesis of the first listed statement: the program was directed to search for a characterization of these small groups. Colton does not report any attempt to use HR to find any structural characterizations that would address any open mathematical problem.

Colton's HR, like Lenat's programs AM and Eurisko, can be used to explore mathematical domains on its own, without direction: they can invent concepts,

[6]A number n is *refactorable* if it is divisible by the number of its divisors. This concept was rediscovered by HR—the numbers were originally called *tau numbers*.

find examples and produce new statements relating those concepts and to prove or disprove them.

A key reason for the conjecture-making success of Fajtlowicz's Grafitti is that the program is likely to produce bounds for invariants for which mathematicians desire bounds. That is, by the very design of the program, it produces statements of a form that might advance our mathematical goals. Recent versions of the program allow the user to specify the invariant for which bounds are sought. In earlier versions of the program though, the produced statements were inequalities between terms which were functions of the invariants known to the program—and many of which were invariants for which mathematicians seek bounds. Thus, even for early versions of the program, where a produced relation may or may not have been an explicit bound for one of these invariants, it was likely to have produced an implicit bound.[7]

One reason that Wang's Program II did not produce any conjectures is that his program was not designed to produce statements that are relevant to our existing mathematical questions and problems. Similarly, Lenat's programs AM and Eurisko were not (explicitly, or even implicitly) designed to produce statements relevant to our existing questions. Colton's HR too, can be run without direction, without an eye to our open problems. It is possible that a program not specifically designed to produce statements relevant to our existing questions might produce such statements—but it would not be expected—we would expect the production of such statements to be the exception rather than the norm.

While such programs may produce (research) conjectures and may discover new theorems, what is ignored that mathematics is a collaborative practice whose advancement is constantly guided by the results and discoveries and concepts of our colleagues. Mathematicians do not ordinarily work in total isolation and, when they do, they would be expected to be much less successful than those aware of the research and advancement and ideas of their predecessors and colleagues. This also partly explains the success of Fajtlowicz's Graffiti: the program does not operate in isolation—many of the invariants that appear in the statements it produces are ones for which mathematicians have an existing interest. If the program were equipped with strictly novel invariants, that is, ones for which there is no existing mathematical interest for bounds for them, it is difficult to see how the produced statements would be of any mathematical interest or that the program would have had the success it has had. In general, it is impossible to say that a program researching a mathematical domain on its own, uninformed of the concepts that our existing mathematical problems involve, would not or could not make (research) conjectures and discoveries. But it would not be expected; to do so it would need to rediscover the concepts involved in our existing mathematical problems and produce statements, involving these concepts, that addressed these problems.

The most recent attempt at automating the production of conjectures is a program due to Gilles Caporossi and Pierre Hansen, colleagues at GERAD. This project by their work that followed the invention, by Hansen and Mladenović [**39**], and development of the Variable Neighborhood Search (VNS) meta-heuristic for finding objects which are extremal (or near-extremal) with respect to an invariant. Caporossi, Hansen and various collaborators have used the results of these heuristic

[7]Fajtlowicz, it should be noted, does not agree with this explanation for the conjecture-making success of his program.

searches to make a number of conjectures. Their results are discussed in a number of publications, the most central of which is their series of papers, "Variable neighborhood search for extremal graphs." [**12, 13, 10, 7, 9, 8, 37, 38**] While the idea behind their VNS heuristic is general, its primary application has been to finding extremal graphs. VNS search is one of the features implemented in their program AutoGraphiX (AGX) for graph-theoretic research. (Caporossi is responsible for the actual development of the program). The program also incorporates techniques for automating conjectures. They first discussed these techniques in 1999 [**11**]. The most complete discussion is in [**13**].

Caporossi and Hansen's papers contain roughly 50 conjectures that are connected to their use of AGX. In some instances, they have used their program to find graphs which are extremal with respect to one parameter (with constraints on one or more other parameters). They then plot the values of this parameter against values of the constrained parameters for these extremal graphs. It may then be the case that there is a clear relationship between these parameters which suggests a conjecture. In this way, for example, they searched for graphs with minimum *energy*[8] E while constraining variously the number of vertices n and the number of edges m of the graphs. Visualizing the relationships between these parameters, patterns emerged and suggested conjectures such as the following:

$$E \geq 2\sqrt{m}$$

$$E \geq \frac{4m}{n}$$

$$E \geq 2\sqrt{n-1} \text{ [7, p. 990]}$$

In other cases, rather than use AGX to produce the data on which they base their own conjectures, the program produces conjectures on its own. In one clearly attributed example, they set AGX to find *color-contrained* trees with minimal *index*.[9] AGX found the following relationship:

$$2\alpha - m - n_1 + 2r - D = 0, \text{ [13]}$$

where α is the independence number of the graph, m is its number of edges, n_1 is its number of pendant vertices, r is the radius, and D is the diameter.

Caporossi and Hansen have proposed two techniques for the automation of mathematical conjecture-making as well as one for automatedly strengthening existing conjectures (or theorems). Perhaps 12 of the 50 conjectures produced with the use of AGX were automatedly produced. In most instances Caporossi and Hansen are not clear about the genesis of the conjectures they report—which makes it difficult to evaluate the success of AGX's conjecture-making heuristics.

The first method they propose is AGX's *numeric* method. This is the method used to produce the aforementioned relation for color-constrained trees. The idea underlying this method is to find some number of examples of objects of a specified class—for instance, color-constrained trees with minimal index—and values of some

[8]The energy of a graph is defined to be the sum of the absolute values of its eigenvalues (of the eigenvalues of its adjacency matrix).

[9]Trees are connected bipartite graphs and thus have a unique bipartition (A, B). *Color-constrained* trees are classes of trees where $|A|$ and $|B|$ are fixed. The *index* of a graph is its largest eigenvalue (the largest eigenvalue of its adjacency matrix).

number of invariants x_1, x_2, ..., x_n for these objects. They then find a basis of all affine relations, that is, relations of the form

$$a_1x_1 + a_2x_2 + \ldots + a_nx_n = b,$$

where the a_i's and b are constants. Should any affine relations be found, they hold for the considered objects and, it is then conjectured, the relations hold for all the objects of the specified class.

AGX does not seem to have produced many conjectures using its numeric method—but this is also not terribly surprising. At least for the case of graphs, affine relations of graph invariants are relatively rare in the literature.

The second method AGX is equipped with for automatedly producing conjectures is its *geometric* method. Here some examples of objects of a specified class are considered, together with values of some number of invariants x_1, x_2, ..., x_n for these objects. AGX then computes

> the convex hull of the corresponding points in the p-space of invariant values. Then each facet of the convex hull provides a conjecture of the form $a_1x_1 + a_2x_2 + \ldots + a_nx_n \geq b$ where the a_j and b are constants.

Caporossi and Hansen report that, applying the geometric approach to find bounds on the *Randic index*[10] R for the class of graphs with maximum degree four, in terms of the number of edges m of the graph and number of pendant vertices n_1, AGX found that

$$R \geq \frac{1}{4}(n_1 + m).$$

This inequality holds for the graphs the program considered and is conjectured to be true for all graphs of maximum degree four.

AGX's geometric method for finding conjectures does not have the limitation of its numerical method—that affine relations between graph invariants are rare. It would seem that AGX should be extremely prolific using its geometric method of conjecture production. Yet AGX seems to have produced only a few conjectures using this method. There are a limitless number of classes of graphs and invariants for which bounds are desired with respect to those classes. What would be of great scientific value in demonstrating the utility of this approach is lists of conjectures produced by the program using AGX's geometric method in those cases of obvious interest to graph theorists, say, for instance, in finding bounds for NP-hard invariants. This would give a much better indication of what to expect using this method to automate the production of conjectures.

AGX's *algebraic* method for strengthening conjectures and theorems begins with an existing inequality, having the form $t \geq 0$, about the relations between the invariants of some class of objects (this relation may be a conjectured one or a theorem). t is some function of primitive invariants x_1, x_2, ..., x_n. AGX first searches for examples of objects that minimize t. The program would then use a database of properties of the given class of objects and test if any of these properties hold for each of the extremal examples that the program has produced (in the case of graphs, these properties might include being a path, a tree, complete, regular,

[10]The Randic index is defined to be the sum of the weights of the edges of the graph, where the weight of an egde vw of the graph, incident to vertices v and w, having degrees $d(v)$ and $d(w)$ is $\frac{1}{\sqrt{d(v)d(w)}}$.

a star, &c.). Suppose the extremal objects have property P and that there is a formula for invariant x for objects having property P. Suppose t is a function of x (as well as possibly other invariants). Let s be formed by replacing x with the known formula. Then AGX produces the conjecture $t \geq s$.

For instance, beginning with Graffiti's conjecture that, for all connected graphs,

$$1 + R - r \geq 0,$$

where R is the Randic index and r is the radius of the graph, AGX found that the graphs which minimize this quantity are paths with two or more vertices. For these paths it is known that $r \leq \lfloor \frac{n}{2} \rfloor$ and that $R = \sqrt{2} + \frac{n-3}{2}$. AGX then produced the strengthened conjecture that, for graphs with two or more vertices,

$$1 + R - r \geq 1 + \frac{n-3}{2} + \sqrt{2} - \lfloor \frac{n}{2} \rfloor, \ [\mathbf{13}]$$

where n is the order of the graph. Hansen and Caporossi also report that AGX was automatedly able to in this way strengthen six other conjectures of Graffiti.

Caporossi and Hansen's methods are, in principle, domain-independent. They apply to any objects that can be represented to a computer. Graphs happen to be the specific case they have used in developing their ideas. Their VNS heuristic has been used with some success—it is clearly a fruitful tool for the investigation of graphs and other objects. Their ideas for automating conjecture-making are interesting but difficult to evaluate—more examples of clearly attributed, automatedly produced conjectures are needed in order to evaluate their fecundity and to judge how well they might serve us in achieving our mathematical goals.

5. Looking Ahead

Research on mathematical conjecture-making is relatively young. Its successful techniques have yet to be extended broadly in mathematical research. But conjecture-making is a central part of mathematical practice. Thus we envision a future where conjecture-making programs will be constant collaborators with mathematicians in all areas of mathematical research, where conjecture-making programs are as widespread and commonly used as computational and visualization environments like Maple and Mathematica are today. Significant steps have now been made towards this goal.

There are three salient points that researchers interested in the existing literature on automated mathematical discovery—that is, on developing programs that actually participate in the advancement of mathematics—should keep in mind. First, the literature is confusing. There are researchers who share the same vocabulary but not the same goals: researchers who use the terms "conjecture" and "discovery" in almost unrelated ways, researchers more interested in the computerized "simulation" of various human behaviors more than in the design of programs having various human abilities, and researchers more interested in designing computers that exhibit various abilities regardless of whether they would be said to simulate those abilities. Second, Fajtlowicz's Principle of the Strongest Conjecture, to produce the strongest statement for which no counterexample is known, has proved to be extremely fecund for finding conjectured bounds of invariants and is presumably extendable to finding conjectures of other forms. Third, successful conjecture-making programs should be designed to produce statements that

address existing research goals. This stands to reason—a program designed to produce statements for reasons that are not explicitly mathematical cannot be expected to produce statements that can contribute to the advancement of mathematics.

One simple idea for extending Fajtlowicz's Principle of the Strongest Conjecture is for making conjectures about relations between classes of objects, finding necessary conditions for an object to have some property P, and thus characterizing the class of objects having property P. Conjectures about the necessary conditions for an object to have property P have the form, "If an object has property P then it has property Q"—saying, in other words, that class of objects having property P is a subclass of the objects having property Q.

Suppose $O_1, O_2, \ldots, O_n$ are (representations of) objects stored in the computer's memory. A property of these objects is a function which associates a truth value ("true" or "false", 1 or 0) to each object of a given class. Let $T_1, T_2, \ldots, T_r$ be computable properties (for a given object O, $T_i = T_i(O)$). Let $f_1, f_2, \ldots, f_s$ represent truth-functional operators (representing "and", "or", "if ... then," &c.). Any term, for instance, $f(T_1, T_2)$ represents a new property. Statements can then be formed from relations of these terms. Suppose necessary conditions for an object to have property P are desired. If Q is such a term, the expression $P \subseteq Q$—which should be interpreted as the statement, "For every object O (of the type of object under consideration), if O has property P then O has property Q"—is a candidate for a conjecture.

The conjectured relations are to be culled from the stream of relations $P \subseteq Q_1$, $P \subseteq Q_2$, $P \subseteq Q_3$, &c. Given a statement of the form $P \subseteq Q$ and a (possibly empty) database of pre-existing conjectures of similar form, $P \subseteq R_1$, $P \subseteq R_2$, ..., $P \subseteq R_l$—the program first checks if the statement "$Q \subset R_1$ and $Q \subset R_2$ and ... and $Q \subset R_l$" is true. That is, check if there is at least one of the objects O from the set $O_1, \ldots, O_n$ for which O does not have property Q but for which it does have properties R_1, R_2, &c. (that is, the the class of stored objects having property Q is a proper subclass of the intersection of the classes having properties R_i). If so, then, with respect to the objects stored in memory, the relation $P \subseteq Q$ says something informative—that is, the relation says something that was not implied by the totality of the previous conjectures of that form that had been kept in the program's database. This is Fajtlowicz's Dalmation heuristic applied in this new setting.

The truth of the relation with respect to the stored objects would then be tested. If the relation is true of all of these objects then it is added to the database of conjectures; and if the relation is false for any of these objects then the general statement that the relation of term functions represents (a relationship between classes of objects having certain properties) is false—and the relation is not accepted as a conjecture. Conjecturing sufficient conditions for an object to have property P would be similar.

Researchers in Artificial Intelligence have attempted to identify general abilities that underly intelligent human behavior. The ability to make conjectures would seem to be a very good candidate for further research.

What follows is a proposal for testing the utility of conjecture-making programs in the design and construction of machines that interact with the physical world. Suppose a computer—call it *Charly*—is connected to a simple mechanical arm whose range of motion is just vertically up and down and which is equipped

with an internal clock and exactly two sensors: one indicating that something is sitting on its "hand," and a second which reports the relative height of the arm. Charly's task is to estimate the weights of objects placed on its hand. The motivating idea here is that a machine can only be equipped with a limited number of special-purpose sensors—some quantities must be estimated from other data. Here Charly is equipped with a clock and two sensors—a more elaborate machine will be equipped with more—and the more it is equipped with the more elaborate will be the variety and richness of the quantities it can conjecture estimates of and, subsequently, use to make decisions about future actions. Ultimately, we would like to expand Charly's capabilities without equipping it with specialized mechanisms for every new task. In particular, we assume its arm is not equipped with a dedicated, built-in, scale.

Charly might estimate the weight of objects by making conjectures about the weights of objects and appealing to these conjectures when some new object is placed on its hand: It represents anything that has triggered its hand sensor for some length of time uninterruptedly as a "physical object." It maintains a database of certain of the physical objects it has encountered: this database consists of a table pairing each object with three numbers corresponding to the three invariants *height*, *time*, and *weight*. Charly has built-in mechanisms for determining the values of the first two of these. The *height* of an object is the distance Charly can raise it from its lowest arm position (using the relative height sensor). The *time* of an object is the amount of time it takes Charly to raise the arm to this height (using the internal clock). Charly has no built-in mechanism for determining the *weight* of an object. It can use its database to make and improve a list of conjectures of the form $weight \leq f(height, time)$. Charly can use its conjectures to make guesses about the weights of objects. When Charly makes a guess that is not sufficiently good, it can be informed of its mistake and the correct weight of the object.

Note that we are in Charly's shoes—we do not have access to an accurate internal scale—we make conjectures about weights of objects and *use* these conjectures in our judgements: the simplest example would be if we were to be given a lump of ore to weigh, a balance and a set of accurate weights. We "weigh" the lump in our hand, perhaps raising and lowering it a few times, and then guess what the first weight to put on the balance should be. That guess may or may not be correct.

The utility, then, of Charly's guesses of the weights of objects would best be measured relative to whatever it uses those guesses for. Perhaps Charly passes the object on to a catapult which uses Charly's guess in calcuating how much force to use to launch it accurately at a barricade. So long as the uses made of Charly's guessed weights have outcomes within some tolerance, there is no reason to check the accuracy of Charly's measurements.

If Charly is informed of an inaccurate measurement as well as the true weight of the object and this weight contradicts one or more of its conjectures, it can add the object to its database of objects, delete the falsified conjectures from the stored list of conjectures, and then proceed to make new conjectures (and, it is hoped, better guesses in the future). If Charly demonstrates any success at guessing weights, this ability can be used as the basis for endowing it with more complex abilities.

References

[1] M. Aouchiche, G. Caporossi, and P. Hansen, Variable neighborhood search for extremal graphs 8: Variations on Graffiti 105, *Congressus Numerantium* 148 (2001) 129-144.

[2] R. Bagai, V. Shanbhogue, J. M. Zytkow, and S. C. Chou, Automatic Theorem Generation in Plane Geometry, in: *Methodologies for Intelligent Systems: 7th International Symposium ISMIS*, Trondheim, Norway (1993) 415–424.

[3] B. Bollobas and P. Erdös, Graphs of Extremal Weights, *Ars Combinatoria* 50 (1998).

[4] J. Brown, and D. Lenat, Why AM and EURISKO Appear to Work, *Artificial Intelligence* 23 (1984) 269–294.

[5] A. Bundy and S. Colton, HR: Automated Concept Formation in Pure Mathematics, in: *Proceedings of IJCAI* (1999) 786–793.

[6] A. Bundy, S. Colton, and T. Walsh, HR: A System for Machine Discovery in Finite Algebras, in: *Proceedings of the Machine Discovery Workshop*, Brighton (1998).

[7] G. Caporossi, D. Cvetkovic, I. Gutman, and P. Hansen, Variable neighborhood search for extremal graphs 2: Finding graphs with extremal energy, *J. Chemical Information and Computer Sciences* 39 (1999) 984–996.

[8] G. Caporossi, P. Hansen, D. Cvetkovic, and S. Simić, Variable Neighborhood Search for Extremal Graphs 3: On the Largest Eigenvalue of Color-constrained Trees, *Linear and Multilinear Algebra* 2 (2001) 143–160

[9] G. Caporossi, P. W. Fowler, P. Hansen, and A. Soncini, Variable Neighborhood Search for Extremal Graphs 7: Polyenes with maximum HOMO-LUMO gap, *Chemical Physics Letters* 342 (2001) 105–112.

[10] G. Caporossi, I. Gutman, and P. Hansen, Variable neighborhood search for extremal graphs 4: Chemical trees with extremal connectivity index, *Computers and Chemistry* 23 (1999) 469–477.

[11] G. Caporossi and P. Hansen, Finding Relations in Polynomial Time, in: *Proceedings of the IJCAI-99* 2 (1999) 780–785.

[12] G. Caporossi and P. Hansen, Variable neighborhood search for extremal graphs 1: The AutoGraphiX system, *Discrete Mathematics* 212 (2000) 29–44.

[13] G. Caporossi and P. Hansen, Variable Neighborhood Search for Extremal Graphs 5: Three ways to automate finding conjectures, Discrete Mathematics (to appear).

[14] F. Chung, The Average Distance and the Independence Number, *Journal of Graph Theory* 12(2) (1988) 229–235.

[15] S. Colton, Refactorable Numbers—a Machine Invention, *Journal of Integer Sequences* (on the WWW at: www.research.att.com/ njas/sequences/JIS) 2 (1999).

[16] E. DeLaVina, Graffiti.pc, *Graph Theory Notes of New York* XLII (2002) 26–30.

[17] S. Epstein, On the Discovery of Mathematical Theorems, in: *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, Milan, Italy (1987) 194–197.

[18] S. Epstein, Learning and Discovery: One System's Search for Mathematical Knowledge, *Computational Intelligence* 4 (1988) 42–53.

[19] S. Epstein, On the Discovery of Mathematical Concepts, *International Journal of Intelligent Systems* 3(2) (1988) 167–178.

[20] P. Erdös, J. Spencer and J. Pach, On the Mean Distance Between Points of a Graph, *Ars Combinatoria* (1986).

[21] P. Erdös, S. Fajtlowicz and W. Staton, Degree Sequences in Triangle-free Graphs, *Discrete Mathematics* 92 (1991) 85–88.

[22] S. Fajtlowicz, A Characterization of Radius-Critical Graphs, *Journal of Graph Theory* 12(4) (1988) 529–532.

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

[24] S. Fajtlowicz, On Conjectures of Graffiti, II, *Congressus Numerantium* 60 (1987) 189–197.

[25] S. Fajtlowicz, On Conjectures of Graffiti, III, *Congressus Numerantium* 66 (1988) 23–32.

[26] S. Fajtlowicz, On Conjectures of Graffiti, IV, *Congressus Numerantium* 70 (1990) 231–240.

[27] S. Fajtlowicz, On Conjectures of Graffiti V, in: *Graph Theory, Combinatorics, and Algorithms: Proceedings of the Seventh Quadrennial Conference on the Theory and Application of Graphs*, Western Michigan University, vol. 1 (1995) 367–376.

[28] S. Fajtlowicz, Toward Fully Automated Fragments of Graph Theory, *Graph Theory Notes of New York* XLII (2002), 18-25.

[29] S. Fajtlowicz, P.W. Fowler, H. Hansen and K.M. Rogers, C60Br24 as a Chemical Illustration of Graph Theoretical Independence, *Journal of the Chemical Society, Perkin Transactions* 2 (1998) 1531–1533.
[30] S. Fajtlowicz and C. E. Larson, Graph-Theoretic Independence as a Predictor of Fullerene Stability, *Chemical Physics Letters* 377/5-6 (2003) 485–490.
[31] S. Fajtlowicz, T. McColgan, T. J. Reid, and W. Staton, Ramsey Numbers of Induced Regular Subgraphs, *Ars Combinatoria* 39 (1995) 149–154.
[32] S. Fajtlowicz and W. Waller, On Two Conjectures of Graffiti, *Congressus Numerantium* 55 (1986) 51–56.
[33] P. W. Fowler, Fullerene Graphs With More Negative Than Positive Eigenvalues; The Exceptions That Prove The Rule of Electron Deficiency, *Journal Chem. Soc. Faraday* 93 (1997) 1–3.
[34] J. Griggs and D. Kleitman, Independence and the Havel-Hakimi Residue, *Discrete Mathematics* 127 (1994) 209–212.
[35] J. Hadamard, *The Psychology of Invention in the Mathematical Field* (Dover Publications, New York, 1954).
[36] G. H. Hardy *A Mathematician's Apology* (Cambridge University Press, Cambridge, 1967).
[37] P. Hansen and H. Mélot, Variable Neighborhood Search for Extremal Graphs 6: Analyzing Bounds for the Connectivity Index, *J. Chem. Inf. Comp. Sci.* 43 (2003) 1–14.
[38] P. Hansen and H. Mélot, Variable Neighborhood Search for Extremal Graphs 9: in: S. Fajtlowicz, P. W. Fowler, and P. Hansen, eds., *Graphs and Discovery* (DIMACS Series in Discrete Mathematics and Computer Science, to appear).
[39] P. Hansen and N. Mladenović, Variable Neighborhood Search, *Computers and Operations Research* 24 (1997) 1097–1100.
[40] K. Haase, Discovery Systems, Technical Memo, AIM-898, MIT Artificial Intelligence Laboratory (1986).
[41] A. Kotlov and L. Lovasz, The Rank and Size of Graphs, *Journal of Graph Theory* 23(1) (1996) 185–189.
[42] C. E. Larson, Intelligent Machinery and Mathematical Discovery, *Graph Theory Notes of New York* XLII (2002) 8–17.
[43] D. Lenat, AM: Discovery in Mathematics as Heuristic Search, in: R. Davis and D. Lenat, eds., *Knowledge-Based Systems in Artificial Intelligence* (McGraw-Hill, New York, 1982) 1–225.
[44] D. Lenat, Artificial Intelligence, *Scientific American*, Sept. 1995, 80–82.
[45] D. Lenat, Automated Theory Formation in Mathematics, in: W. Bledsoe and D. Loveland, eds., *Automated Theorem Proving: After 25 Years* (American Mathematical Society, Providence, 1984) 287–314.
[46] D. Lenat, Computer Software for Intelligent Systems, *Scientific American*, Sept. 1984, 204–213.
[47] D. Lenat, Eurisko: A Program that Learns New Heuristics and Domain Concepts, *Artificial Intelligence* 21 (1983) 61–98.
[48] D. Lenat, On Automated Scientific Theory Formation: A Case Study Using the AM Program, *Machine Intelligence* 9 (1979) 251–283.
[49] D. Lenat, The Nature of Heuristics, *Artificial Intelligence* 19 (1982) 189–249.
[50] D. Lenat, The Role of Heuristics in Learning by Discovery: Three Case Studies, in: R. Michalski, J. Carbondell, and T. Mitchell, eds., *Machine Learning* (Morgan Kaufmann, Los Altos, CA, 1983) 243–306.
[51] D. Lenat, Theory Formation by Heuristic Search, *Artificial Intelligence* 21 (1983) 31–59.
[52] D. Lenat, The Ubiquity of Discovery, *Artificial Intelligence* 9 (1978) 257–285.
[53] A. Newell, J. Shaw and H. Simon, Empirical Explorations with the Logic Theory Machine, in: E. Feigenbaum and J. Feldman, eds., *Computers and Thought* (AAAI Press, Menlo Park, 1995) 109–133.
[54] R. Pepper, On New Didactics of Mathematics: Learning Graph Theory via Graffiti, in: S. Fajtlowicz, P. W. Fowler, and P. Hansen, eds., *Graphs and Discovery* (DIMACS Series in Discrete Mathematics and Computer Science, to appear).
[55] G. Ritchie and F. Hanna, AM: A Case Study in AI Methodology, *Artificial Intelligence* 23(3) (1984) 249–268.

[56] H. Wang, Computer Theorem Proving and Artificial Intelligence, in: W. Bledsoe and D. Loveland, eds., *Automated Theorem Proving: After 25 Years* (American Mathematical Society, Providence, 1984) 49–70.

[57] H. Wang, Toward Mechanical Mathematics, in: J. Siekmann and G. Wrightson, eds., *Automation of Reasoning: Classical Papers on Computational Logic 1957–1966* (Springer-Verlag, Berlin, 1983) 244–264.

Department of Mathematics, University of Houston, Houston, Texas 77204
E-mail address: clarson@math.uh.edu