Intelligent Machinery and Mathematical Discovery

C. E. Larson
Department of Mathematics
University of Houston
clarson@math.uh.edu

October 1, 1999

Abstract

All published research on automated mathematical conjecture-making is surveyed, and the ideas underlying the successful programs in this area are outlined. One particularly successful—and little known—program is comprehensively described for the first time. The fundamental principle underlying this program can be simply stated: make the strongest conjecture for which no counterexample is known. Conjecture-making may be key to building machines with a wide variety of intelligent behaviors. If so, this principle should prove exceptionally useful.

1 Introduction

In the early 1980's Edward Fredkin, an engineer and entrepreneur who had once headed M.I.T.'s Laboratory of Computer Science, offered a prize of \$100,000 to the researcher that first constructed a machine that made an important mathematical discovery. This problem belongs to the roughly half-century old field of Artificial Intelligence (AI), a branch of computer science. Fredkin's prize hadn't been awarded by 1995 when Raj Reddy included automated mathematical discovery among the "Grand Challenges of AI." Reddy, it is worth noting, is a recipient of the Turing Award, essentially the Nobel Prize for computer science. He wrote:

> Several exciting problems in AI are seemingly reasonable and yet currently unsolvable. Solutions to these problems will require major new insights and fundamental advances in computer science and artificial intelligence. Such problems include ... the discovery of a major mathematical result by a computer ... [37, p. 301]

Fredkin's prize remains unrewarded. [1]

[1] Fredkin's prize is administered by Carnegie Melon University.

Surprisingly, back in 1956, two of the "founding fathers" of AI, Herbert Simon and Allen Newell—also Turing Award winners—predicted "[t]hat within ten years a digital computer will discover and prove an important new mathematical theorem." [40, p. 7] What has been achieved in automating mathematical discovery? Why has success been so long in coming? In this paper I will survey the research that has been done towards endowing machines with the ability to make mathematical conjectures. It isn't always clear what is meant by a machine that makes mathematical discoveries but, under any interpretation, such a machine would make conjectures.

There are at least four reasons one may be interested in a survey of the research on automated mathematical conjecture-making. One obvious reason is mathematical: one may be interested in knowing how machines can be used to advance certain mathematical goals. Another reason is historical: one may be interested to know dates of various advances, who achieved them, and how quickly research is progressing, among other things. A third reason is that success in constructing a machine with a certain ability may lead to advances in constructing machines with relevantly similar abilities—the survey provides a vehicle to transmit ideas to researchers working outside mathematical conjecture-making but whose problems may be relevantly similar. Lastly, it can help all researchers working to construct machines with abilities exhibiting intelligence identify successful and unsuccessful methodologies—as all researchers must choose how to pursue their goals. Some surveyed research was successful, while some was not. Can anything be said about what the successful researchers did right, or what the unsuccessful ones did wrong?

Mathematics comprises various activities. The end products are familiar: things like calculus books used to train engineers in various abilities useful in their work. The fundamental results in any mathematical text include its theorems, mathematical propositions that have been proved. The existence of a theorem rarely begins as the last line of a proof. Rather a mathematician conjectures that some mathematical proposition is true or it is not. He, or other mathematicians, may try and prove or disprove the proposition. In working on a proof, a mathematician may introduce a new concept in the belief that it will assist his work. Conjecture-making, theorem-proving, and concept formation are three of the central activities in mathematics and the ones researchers have attempted to automate. In order to win Fredkin's prize, the successful machine may require all three of these abilities. By far the most research has been conducted in automated theorem-proving. This work was also the first to begin—in the mid-1950's—and is the only one able to support a journal. Much less work exists in the other two areas.

In automated conjecture-making, perhaps two dozen papers have been published. Researchers in this field have written the following programs. (The researcher's name is followed by the name of the program and then by the year of its first published description.)

D. Lenat, AM, 1975.

D. Lenat, Eurisko, 1977.

K. Haase, Cyrano, 1986.

S. Fajtlowicz, Graffiti, 1986.
S. Epstein, GT, 1987.
R. Bagai et al., unnamed, 1993.
S. Colton et al., HR, 1998.

2 Lenat's AM

In the mid-1970's Douglas Lenat published the first research on automated mathematical conjecture-making: he reported on his work with his program AM (originally an acronym for "Automated Mathematician" [39, p. 229]) and then a successor in roughly a dozen papers between 1975 and 1984. [26, 28, 29, 30, 31, 32, 30, 34, 33, 35]

> [AM] is a computer program which defines new concepts, investigates them, notices regularities in the data about them, and conjectures relationships between them. [28, p. 288]

The word "conjecture" admits various uses: a teacher might call a student's guess that there is a ruler-and-compass construction of the trisection of an angle a "conjecture," even though he knows it is false; 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 that 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—henceforth the term "research conjecture" will be used to distinguish such genuinely novel mathematical advances. Clearly, when Fredkin offered his prize, when Reddy made his challenge, when Simon and Newell made their prediction, what they had in mind was a machine that could advance mathematics, a machine that could make research conjectures.

That Lenat intended AM to make (research) conjectures is clear. He wrote,

> To demonstrate the efficacy of its methods to practitioners of the fields it works in (e.g., mathematicians) and to practitioners of AI, any program claiming to be a "discovery program" should aspire to . . . make at least a few genuinely new (to mankind) useful discoveries. [30, p. 73]

Applied to his own program his statement says: To demonstrate the efficacy of its methods to practitioners of AI, AM should make some mathematical (research) conjectures. Lenat reported that AM failed to do this: "AM was not able to discover any 'new-to-mankind' mathematics purely on its own . . . ." [26, p. 6] AM did output mathematical propositions, including some very well-known theorems and conjectures, such as the Fundamental Theorem of Arithmetic and Goldbach's Conjecture. AM may be said to have made conjectures—depending on how one uses that term—but what it certainly did not do is make any research conjectures.

AM has been widely acclaimed—in 1977 Lenat received the Computers and Thought Award for "the most impressive contribution to the subject [of AI]" in the two years preceding. [35, p. 257n] This acclaim, obviously, was not for writing a program which makes (research) conjectures; he had other aims in writing his program, not relevant to the current discussion, which it presumably accomplished and for which he was recognized. Was his work on automated conjecture-making of any value to future research on the subject?

Consider the claim that Lenat showed that machines *could* be endowed with the ability to make (research) conjectures, as a biographical sketch from a 1995 issue of *Scientific American* suggests:

> He received his Ph.D. in computer science in 1976 from Stanford University with a demonstration that computers could be programmed to propose original mathematical theorems. [27, p. 82]

Under what circumstances can such a claim be made? Obviously, if the program had, in fact, made (research) conjectures. When else? Perhaps retrospectively, in case the ideas underlying his work were subsequently used in constructing a machine that made (research) conjectures. Were they? Almost a quarter-century has passed since his ideas were first published and, in that time, there have been attempts to implement his ideas in other programs. Lenat himself wrote one, Eurisko, a successor to AM. Lenat did not report any (research) conjectures it made, nor did he claim that it made any. He did not attempt any further conjecture-making programs. (In 1984, while the ideas behind the program that led to his fame remained unproven, he obtained massive funding over multiple years to pursue a new idea—and he has not returned to research on conjecture-making programs.) In the mid-1980's Ken Haase, a student of Marvin Minsky (another "founding father" of AI and Turing Award winner) who had access to Lenat, made another attempt to implement the ideas of AM. Haase described his program Cyrano as follows:

> Cyrano is a thoughtful reimplementation of Lenat's controversial Eurisko program, designed to perform automated discovery and concept formation in a variety of technical fields. [24, p. 1]

Haase did not report that Cyrano made any mathematical conjectures—and there is no reason to believe that it did. Given this record of failure, it is hard to imagine grounds for saying that Lenat proved the *possibility* of automated mathematical conjecture-making.

There are practical questions that every researcher in automated conjecture-making faces: should I try to implement any of Lenat's ideas in my own program? Is my program more likely to be successful if I do this or if I try some other approach? What reasons are there to choose one approach or the other? G.D. Ritchie and F. K. Hanna, in their 1984 critique of Lenat's AM research concluded "that it would be extremely difficult to base further research in this area on AM." [38, p. 266] AM's failure and Ritchie and Hanna's critique provide a researcher with reasons to ignore Lenat's ideas. What reasons are there to do otherwise?

AM is an example of a "rediscovery" program, a program that outputs known propositions of science or mathematics, but does not contribute any new propositions of value to research. (A program that does is called a "discovery" program.) There has been a relative explosion of rediscovery programs in the twenty-odd years since AM first appeared. The question every researcher in automated mathematical conjecture-making must address can be generalized to a question every researcher trying to construct a program that makes scientific discoveries must address: should I try and implement the ideas of a given program that never made a discovery, and whose ideas have never been proved, or should I try some other approach?

3 Graffiti

Graffiti, a program conceived by Siemion Fajtlowicz at the University of Houston (and developed, since 1990, with Ermelinda DeLaVina), was the first program to have actually made (research) conjectures. While the first paper on the program appeared more than a dozen years ago, Graffiti still seems to be virtually unknown except among mathematicians (particularly graph theorists)—there does not seem to have been any attempts to implement its ideas in another program. Fajtlowicz's program was first described in his series of papers, "On Conjectures of Graffiti." This survey contains the first self-contained description of the program.

On what grounds can it be said that the propositions Graffiti has advanced are (research) conjectures? In the first place, many, many of these propositions are new. This is a necessary condition. Nevertheless, it is trivial to write a program that forms statements never before considered by anyone: consider a program that outputs statements of the form "n is prime," where n is a randomly chosen thousand digit number. What is a sufficient condition that a new mathematical proposition be a (research) conjecture? One might think it is sufficient that a mathematician does, in fact, investigate it. But a mathematician may investigate a mathematical proposition on a whim—for no better reason than that for attempting a crossword puzzle. Furthermore, the fact that no mathematician has investigated a proposition may be only because none who could recognize its value ever encountered it. One might think that it is sufficient that research inspired by the proposition is published. Publication is certainly strong circumstantial evidence that the proposition is different in character from one asserting the primality of a thousand digit number—nevertheless it is possible that publication was due more to an editor's whimsy than any other reason. The two criteria just considered are matters of mere historical fact: when Fredkin offered his prize, when Reddy proposed his "grand challenge," and when Simon and Newell made their prediction, their aims could not have been that the construction of a machine would lead a human to act in a specified way—their aim was for a machine that contributes to the advancement of mathematics. Many researchers, Fajtlowicz included, have stated that their own aim is that the machine produce "interesting" propositions. What is meant, though, by "interesting"—if it means that it produces a certain emotion in one or more mathematicians, then the fact that a mathematical proposition is interesting cannot be a sufficient condition that it be a (research) conjecture. The most clearcut way in which the proposal of a new mathematical proposition advances mathematics is when it can (if true) help in achieving some existing mathematical aim: for instance, if it contributes an essential step in a proof. I won't claim that this condition is either a necessary or a sufficient one for a proposition to be a (research) conjecture, only that it is the best existing measure with which to judge this question. Consider the statement that a certain huge number is prime. If this proposition does not advance some existing mathematical aim then, on the proposed measure, it is not a (research) conjecture. To return now to our question regarding the value of the propositions advanced by Graffiti, many of its propositions involve bounds for certain numbers which cannot be efficiently computed. These propositions advance the aim of finding better bounds than currently known. These are undoubtedly (research) conjectures.

Graffiti's conjectures have, in fact, instigated mathematical research— circumstantial evidence that the propositions are both new and significant. There are now numerous papers, theses, and dissertations in which these conjectures (or related weaker and stronger propositions) are proved, or for which counterexamples are found.[2] Graffiti has proved to be a genuine contributor to the advance of mathematics: its collaborators include students as well as the best-known graph theorists—including Noga Alon, Bela Bollobas, Fan Chung, Paul Erdös, Jerry Griggs, Daniel Kleitman, Laszlo Lovasz, Paul Seymour and Joel Spencer [2, 6, 11, 12, 23, 25, among others]—who hail from the most renowned centers of learning in the world to its most obscure—from Cambridge and Princeton Universities to the Heilongiang Water Conservancy College and the Lianzhou Railway Institute. [36]

Graffiti's first (research) conjectures were in the field of graph theory. Its underlying ideas, as described in Fajtlowicz's papers [16, 18, 20, in particular], apply not just to graphs: Graffiti has also made conjectures in geometry, number theory, and even chemistry[3]—conjectures about the structure of fullerenes (as represented by their graphs) have already led to at least one paper by the fullerene expert Patrick Fowler. [22, 13] Graffiti's underlying ideas can be applied to any objects, mathematical or otherwise, that can be represented by a computer—that is, its fundamental ideas are domain independent. What follows is a description of Graffiti's operation sufficiently fine so that researchers in other areas should be able to apply its ideas and test their usefulness.

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 this type of object is a property of the objects which is independent of the representation of the objects. Some invariants correspond to numbers. Let $\alpha_1, \alpha_2, \ldots, \alpha_n$ be computable numerical invariants (for a given object O, $\alpha_i = \alpha_i(O)$). Let $f_1, f_2, \ldots, f_s$ represent proce-

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

[3]The entire list of conjectures is available on the WWW at: math.uh.edu/˜clarson.

dures instantiating the operations of the algebraic system (these might include, for instance, "plus," "times," &c.) Any term, like $f(\alpha_1, \alpha_2)$—or concretely, $\alpha_1 + \alpha_2$—represents a new numerical 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. Suppose conjectures about the upper bound 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 possible conjectures. [20, pp. 370-371] Given a statement of the form $t \leq s$ and a (possibly empty) database of pre-existing conjectures of similar form, $t \leq u_1$, $t \leq u_2$, $\ldots$, $t \leq u_l$—the Dalmatian heuristic checks if the statement "$s(O) \leq u_1(O)$ and $s(O) \leq u_2(O)$ and $\ldots$ and $s(O) \leq u_l(O)$" is true for at least one of the objects O from the set $O_1, \ldots, O_n$. If it is then, with respect to the objects stored in memory, the relation $t \leq s$ 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—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 a counterexample is not 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. [17, 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. [18, p. 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.

It is of the essence of a conjecture that it may be false and so it may be with the statements that Graffiti produces. 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 employs various techniques to speed it along. One is to keep the database of objects relatively small—in fact it only stores (representations of) objects which it has found informative, that is, which have served as a counterexample to a conjecture. Another is to keep the database of conjectures relatively small. 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). These 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.

We turn now from describing Graffiti's operation to giving an example of the concrete objects it typically considers, a concrete invariant, and one of its actual conjectures. The majority of the program's conjectures have been about graphs (a graph is a collection of *vertices* together with a set of *edges* joining some or all of the vertices). In this case the *objects* are graphs—these may be represented in a computer as 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 cardinal of the largest set of vertices where no two are joined by an edge. This invariant is of enormous practical importance and, while it can be computed, it can't be computed efficiently (as a function of the number of vertices of the graph it takes, in general, an exponential amount of time)—so conjectures about its upper and lower bounds represent possible important new knowledge. If the independence number of a graph is represented by the letter α, 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—and the topic of one of the first papers published about any computer-generated conjecture—was it's Conjecture No. 2

regarding the lower bound of the independence number: "(For every connected graph) average distance (between any two vertices in the graph) $\leq$ independence number (of the graph)." (The conjecture was proved by Fan Chung. [6])

The question was raised earlier whether a researcher writing an automated conjecture-making program should try and implement Lenat's ideas. Reasons now exist to ignore two of his specific claims: that conjecture-making programs require huge numbers of heuristics and that new heuristics are required for every domain in which a program makes conjectures. Fajtlowicz's work on Graffiti has shown that neither is necessary for a program to make conjectures.

AM employed a whopping 250 heuristics yet, in Lenat's view, AM's failure was due in part to a lack of heuristics:

> AM's key deficiency appeared to be the absence of heuristics which cause the creation and modification of new heuristics. [34, p. 57]

No version of Graffiti has employed more than four heuristics. Graffiti's success demonstrates that conjecture-making programs do not require large numbers of heuristics much less heuristics for finding heuristics.

Lenat also claimed that a requirement for programs which can make conjectures in multiple domains is that it employ domain specific heuristics. The following statements are among the reasons behind his claim.

> [I]n distinct fields of science and mathematics ... [n]ot only are the concepts different, so are most of the powerful heuristics. [28, p. 288]
>
> Working in point-set topology with geometry heuristics is not very efficient, nor was AM's working in number theory using only heuristics from set theory. [32, p. 23]

Graffiti has made conjectures in graph theory, number theory, geometry and chemistry, using only the domain-independent heuristics described earlier. This disproves Lenat's claim.

Lastly, I will make a claim of my own: I claim that the ideas behind Graffiti are of use in many areas of artificial intelligence research, for instance, in constructing machines that interact in certain ways with the physical world. Graffiti has only made conjectures about abstract objects, there is no a priori reason why a computer adhering to Graffiti's general principles could not make conjectures about physical objects. Consider the following thought experiment. Suppose a computer—call it *Charly*—was connected to a simple mechanical arm whose range of motion is just vertically up and down (Fig. 1) and which is equipped with an internal clock and the following two sensors: one indicating that something is sitting on its "hand," and a second which reports the relative height of the arm. Suppose one of Charly's tasks require it to have a reasonable estimate of the weights of certain objects resting on its hand. Of course, a scale could be built in to its arm. But Charly may have many responsibilities and we prefer to construct it with a minimum of specialized mechanisms—utilizing more general mechanisms whenever there is evidence that this can be done.

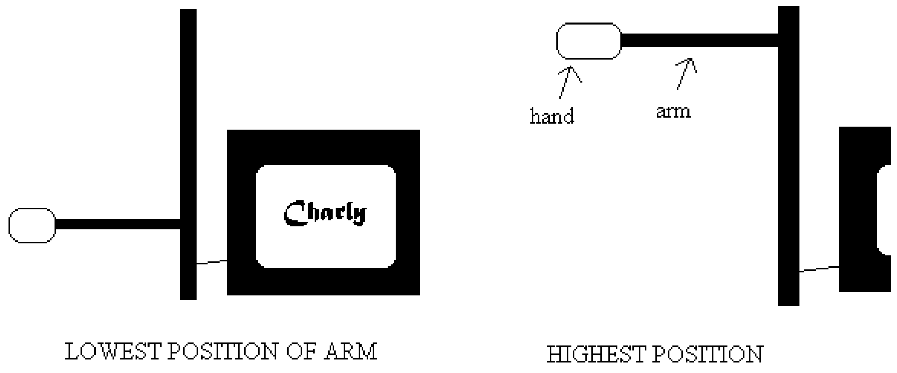

Figure 1: Charly weighs the situation

Charly might implement the task as follows. 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 a row of three numbers, numbers corresponding to the three invariants *height*, *time*, and *weight*. Charly has procedures 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 procedure 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 for its task, it can be informed of its mistake and the correct weight of the object. If this weight contradicts one or more of Charly's conjectures it can add the object to its database of objects, delete the falsified conjectures from the 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 constructing it with more complex abilities.

4 Other Programs, Other Ideas

This last section contains accounts of three programs which have appeared since Graffiti did.

Susan Epstein's Graph Theorist (GT) [8, 9, 10] appeared in 1987. She described it as follows:

> The Graph Theorist, GT, is a knowledge-intensive, domain-specific learning system which uses algorithmic class descriptions to discover and prove relations among mathematical concepts . . . Mathematical discovery includes the creation of new mathematical concepts, the conjecture of relations among concepts and the proof or disproof of such conjectures. [8, p. 194]

Epstein claimed her program made conjectures. Mathematicians would assume she means that it made what have, in this paper, been called "research conjectures"—but none of the four examples that she reports from GT's output are new mathematical propositions. The first, for instance, is "Every tree is acyclic"—which can be found in any graph theory text. There is no evidence that GT made any (research) conjectures. (Epstein, as did Lenat, pursued other aims besides automated conjecture-making—and no general conclusions should be inferred from this report.)

Perhaps because the field of Artificial Intelligence is young—and successful practices require time to be developed—published work on the mechanization of a given ability often redescribes not only the mechanisms which have demonstrated success in past attempts but even those which have remained unproven. A survey of cancer treatments, in contrast, will not include descriptions of "Psychic Surgery" or of homeopathic methods. They are unproven. It might be thought that there is a difference between these cases: that in the medical case there is no theoretical reason why the fringe treatments *should* work—while in the AI case there are. This is not true, at least, in the subfield which aims to construct conjecture-making machines or, more generally, machines with abilities never before mechanized. There is no theory of the construction of such machines. There are only methods which have had some success in mechanizing the desired ability, and methods which have proven successful in mechanizing relevantly similar abilities.

In 1993 an unnamed program was described in a paper by colleagues from Wichita State University: R. Bagai, V. Shanbhogue, J. M. Zytkow, and S. C. Chou. [1] This program formed mathematical propositions but only advanced them if it also had a proof for them. These researchers gave the following description of their program:

> Our mechanism incrementally generates geometrical situations, makes conjectures about them, uses a geometric theorem prover to determine the consistency of situations, and keeps valid conjectures as theorems. [1, p. 415]

They give only two examples of the theorems output by their program, neither of which was a new mathematical proposition: the first of these was that the diagonals of a parallelogram intersect, and the second was Euclid's Parallel Postulate. This is no evidence that their program contributes to mathematical research.

The last program accounted for in this survey—the most recent to appear—is HR (for "Hardy-Ramanujan"), a program which does advance new mathematical propositions. It was created by researchers, headed by Simon Colton, at the University of Edinburgh and their research is still in progress. [4, 5, 7] They describe their program as follows:

> The HR program forms concepts and makes conjectures in domains of pure mathematics and uses theorem prover OTTER and model generator MACE to prove or disprove the conjectures. [4]

The HR team is extremely ambitious and the automation of conjecture-making is only one of their numerous aims.

The papers describing HR contain roughly a dozen examples of the new mathematical propositions that the program has produced. The following three propositions (new to me) are representative of its output; they are definitely credited to the program (which is sometimes unclear). They 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, $f(G) = |\{(a, b, c) \in G^3 : a * b = c \textit{ and } b * c = a\}|$ [5]

2. No perfect number is refactorable. [7]

3. The refactorable numbers are a subsequence of the sequence of positive integers congruent to 0, 1, 2, or 4 (mod 8). [7]

Do these propositions contribute to the advancement of mathematics? That is, are they (research) conjectures? Could they aid in advancing some pre-existing mathematical goal? The first proposition is obviously trivial and, as to the other two, I don't myself know the answer—and Colton's papers are silent on the subject. Note, it is no criticism that the first proposition is trivial: it may still advance our mathematical aims. It may be just the step needed in a proof, for instance—though trivial, no one else may have thought of it. The program is quite new so one can't expect any circumstantial evidence yet regarding the value of the second and third propositions, or of the rest of HR's published production, such as mathematical papers they have inspired. The value of these propositions may become clearer in time.

The fact that Fajtlowicz's program Graffiti is designed to produce inequalities and that finding bounds of certain numbers is an aim of mathematics explains, in part, why the program produces (research) conjectures. HR is not designed to produce inequalities—but mathematical propositions of other forms. An interesting question is: when do propositions of these forms advance the aims of mathematics, and how can a computer be constructed to do this? The three examples of HR's output belong to group theory and number theory. The forms of these propositions are not specific to these branches of mathematics though—in fact, HR's methods are domain independent. Their forms are as follows.

1. An integer-valued function f defines an equivalence relation which corresponds to a specified partition of objects (of a given type).

What classification problems are existing aims in mathematics? Obviously, a new proposal for the solution of an outstanding classification problem will be a (research) conjecture. When does the classification of objects serve as intermediary step in the pursuit of a mathematical goal, for instance, the proof of a theorem? How can a machine be designed to automate the pursuit of such intermediate goals successfully?

2. If an object has property P then it has property Q (for objects of a given type, and where P and Q are represented by first-order formulas in the language of the theory of those objects).

When do conditional propositions advance the aims of mathematics? The most obvious case is when they serve as intermediate steps in proofs of other conditional statements. Suppose an existing aim of mathematics is to prove some conditional proposition. How can a machine be constructed that produces conditional propositions that do advance this aim?

3. One structure is a sub-structure of another (of the same type).

When do these propositions advance the aims of mathematics? How can they be used to advance the proof of a desired proposition? How can this be automated?

Propositions of the three forms discussed here may be weaker or stronger. One wonders, given a program which produces propositions of these forms and which is designed to achieve some clear mathematical aim, whether Fajtlowicz's principle of making the strongest conjecture for which a counterexample is not known will be the useful technique it has proved to be in producing inequalities. Such questions have not yet been addressed in the published research of Colton and the HR team—as mentioned, they have been pursuing a number of goals simultaneously—but it will be interesting to see how they do tackle these questions and what success they achieve.

References

[1] 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.

[2] B. Bollobas and P. Erdös, Graphs of Extremal Weights, *Ars Combinatoria*

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

[4] A. Bundy and S. Colton, HR: Automated Program Formation in Pure Mathematics, in: *Proceedings of IJCAI* (1999) to appear.

[5] A. Bundy, S. Colton, and T. Walsh, HR: A System for Machine Discovery in Finite Algebras, in: *Proceedings of the Machine Discovery Workshop ECAI99*, Brighton (1998) to appear.

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

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

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

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

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

[11] P. Erdös, J. Spencer and J. Pach, On the Mean Distance Between Points of a Graph, *Congressus Numerantium* 63 (1986).

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

[13] 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.

[14] S. Fajtlowicz and W. Waller, On Two Conjectures of Graffiti, *Congressus Numerantium* 55 (1986) 51–56.

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

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

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

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

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

[20] 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.

[21] S. Fajtlowicz, T. McColgan, T. J. Reid, and W. Staton, Ramsey Numbers of Induced Regular Subgraphs, *Ars Combinatoria* 39 (1995) 149–154.

[22] 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.

[23] J. Griggs and D. Kleitman, Independence and the Havel-Hakimi Residue, *Discrete Mathematics* 127 (1994) 209–212.

[24] K. Haase, Discovery Systems, Technical Memo, AIM-898, MIT Artificial Intelligence Laboratory (1986).

[25] A. Kotlov and L. Lovasz, The Rank and Size of Graphs, *Journal of Graph Theory* 23(1) (1996) 185–189.

[26] 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.

[27] D. Lenat, Artificial Intelligence, *Scientific American*, Sept. 1995, 80–82.

[28] 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.

[29] D. Lenat, Computer Software for Intelligent Systems, *Scientific American*, Sept. 1984, 204–213.

[30] D. Lenat, Eurisko: A Program that Learns New Heuristics and Domain Concepts, *Artificial Intelligence* 21 (1983) 61–98.

[31] D. Lenat, On Automated Scientific Theory Formation: A Case Study Using the AM Program, *Machine Intelligence* 9 (1979) 251–283.

[32] D. Lenat, The Nature of Heuristics, *Artificial Intelligence* 19 (1982) 189–249.

[33] 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.

[34] D. Lenat, Theory Formation by Heuristic Search, *Artificial Intelligence* 21 (1983) 31–59.

[35] D. Lenat, The Ubiquity of Discovery, *Artificial Intelligence* 9 (1978) 257–285.

[36] W. Liuxing, Z. Jianxun, and M. Kejie, On Graffiti's Conjecture 585, preprint.

[37] R. Reddy, Grand Challenges in AI, *ACM Computing Surveys* 27(3) (1995).

[38] G. Ritchie and F. Hanna, AM: A Case Study in AI Methodology, *Artificial Intelligence* 23(3) (1984) 249–268.

[39] D. Shasha and C. Lazere, *Out of Their Minds* (Copernicus, New York, 1995).

[40] H. Simon and A. Newell, Heuristic Problem Solving: The Next Advance in Operations Research, *Operations Research* Jan.–Feb. (1958) 1–10.