Computer Aided Serendipity: The Role of Autonomous Assistants in Problem Solving

James Arvo
Department of Computer Science
California Institute of Technology
Pasadena, CA 91125, USA
arvo@cs.caltech.edu

Abstract

Pencil and paper are perhaps the most effective problem-solving tools ever invented. Why is this so, and what does this portend for computer-assisted problem solving? In this paper we investigate why the computer has not made more significant inroads into many aspects of problem solving, even in domains ostensibly concerned with purely formal methods. We observe that for many problem-solving activities computers are currently more constraining than enabling, particularly during problem formulation. We identify some of the obstacles that must be overcome in making the computer a more attractive medium for problem solving, and explore some of the tools that will likely play a part in bringing this about.

Key words: artificial intelligence, autonomous assistants, belief revision, human-computer interaction, numerical experiments, symbolic computation.

1 Introduction

The Oxford American Dictionary defines *serendipity* as "the making of pleasant discoveries by accident." The history of science is filled with examples of serendipitous discoveries, from the invention of nylon to the discovery of pulsars [23]. As one medical researcher recently put it, making important discoveries is more a game of pool than billiards: "You score points regardless of which pocket the ball goes into." He cites the fact that nearly every gene linked to breast cancer was discovered by researchers seeking something quite unrelated [17].

While all discoveries, by their very nature, involve some element of surprise, they cannot be entirely accidental. In fact, the word serendipity, as originally coined by Horace Walpole, carried the connotation of fortuitous accident coupled with the sagacity to recognize its significance. Of course, sagacity needn't be entirely *ex post facto*. Serendipitous discoveries often occur once the search has been painstakingly narrowed to a few promising avenues. Indeed, any rational approach to problem solving leaves as little to chance as possible, in accord with the famous dictum of Louis Pasteur: "In the field of observation, chance favors only the prepared mind." Discoveries that are to some degree foreseen, and result from goal-directed winnowing of the alternatives, are sometimes referred to as "pseudo-serendipitous" [23]. The role of chance is frequently downplayed in such discoveries, if not entirely discounted. Nevertheless, without an element of chance, "discovery" is nothing more than verification; without sagacity, it is mere happenstance.

The focus of this paper is the prospect of computer-assisted discovery, which necessarily entails both elements of serendipity: intelligence in search and recognition, and the possibility of fortuitous accidents. Our emphasis shall be on machines that *assist* human investigators rather than those that seek discoveries independently; although, as we shall see, autonomy is a vital attribute of such a machine. While research in artificial intelligence has demonstrated the feasibility of purely autonomous discovery by machines [18, 34, 20], the potential for computer-assisted discovery appears to be far greater in the near term.

In addition to considering only this more modest aim, we shall further limit the scope of the discussion to a domain in which the computer has already proven to be a valuable parter in exploration: namely, mathematics. Indeed, remarkable progress has been made in areas such as computer algebra [21, 33], mechanical theorem proving [8], and computer-aided construction and verification of proofs [26, 31].

Despite their impressive repertoire, existing tools for computer-aided mathematics are simply not amenable to the haphazard process by which humans make serendipitous discoveries, nor do they accommodate the ill-defined methods by which we *formulate* problems. Unfortunately, the latter often requires the preponderance of the labor. As Barwise and Etchemendy put it, "In problem solving, well begun really is half done" [5]. In the remainder of the paper, we examine these shortcomings in greater detail and suggest avenues by which the computer

may become a more attractive vehicle for human-oriented problem solving.

2 The Nature of the Challenge

Since the 1950's researchers have envisioned the computer as a means of solving all manner of problems that are beyond human abilities. In the early days of artificial intelligence, Ashby conjectured that important socio-economic problems could be resolved by human intellect that was computationally enhanced [4]. The idea behind Ashby's "intelligence amplifier" was to extend human problem solving in two very general ways:

1. Production of possible solutions
2. Selection of solutions that work

Today computers perform both functions in a variety of applications from game playing to artificial life. However, nothing like the general-purpose intelligence amplifier has come to pass. There are two fundamental reasons: 1) enormous obstacles arise in problem formulation, which often requires great ingenuity, and 2) the resulting search is frequently intractable. We first examine problem formulation, which has important implications for human-computer interaction.

A cursory look at the way people solve problems, even in the relatively restricted domain of mathematics, reveals an array of techniques that incorporate diagrams, prose, and symbols with varying degrees of completeness and correctness [5, 25]. As a concrete illustration, Figure 1 shows a typical page from one of the author's notebooks. The page contains fragments of mathematics mingled with annotations and sketches that explain the meanings of symbols and evoke connections with appropriate mathematical machinery; in this case, Stokes' theorem. Problem solving often begins by recording such meanderings in notebooks or on napkins. The nature of these scribbles highlights the gulf between humans and machines, even in the realm of symbolic mathematics. The content of Figure 1 is the very antithesis of a syntactically well-defined language suitable for mechanized processing.

The vast majority of user interfaces today cannot interpret diagrammatic input. Furthermore, malformed, ambiguous, or contradictory information from a user is handled only superficially. Lacking any notion of tentative "scratch work," the typical interface rejects input that is not complete and correct with respect to the current state of the program. At best, the user is prompted to try again. This is one of the many ways in which human-computer interaction differs dramatically from human-human interaction. Shared background knowledge, default assumptions, and an ability to tolerate a certain amount of ambiguity – in hopes that it will be rectified later – are a few of the factors that make communication among people far simpler. Moreover, when confronted with contradictory information, humans exhibit at least a limited capacity to re-examine and revise previous assumptions, and thereby avoid obviously inconsistent beliefs. These attributes are not yet exhibited by computers in their interaction with humans. Consequently, computer-aided design tools are frequently more constraining than enabling, as the user is forced to proceed in a fashion that suits the system.

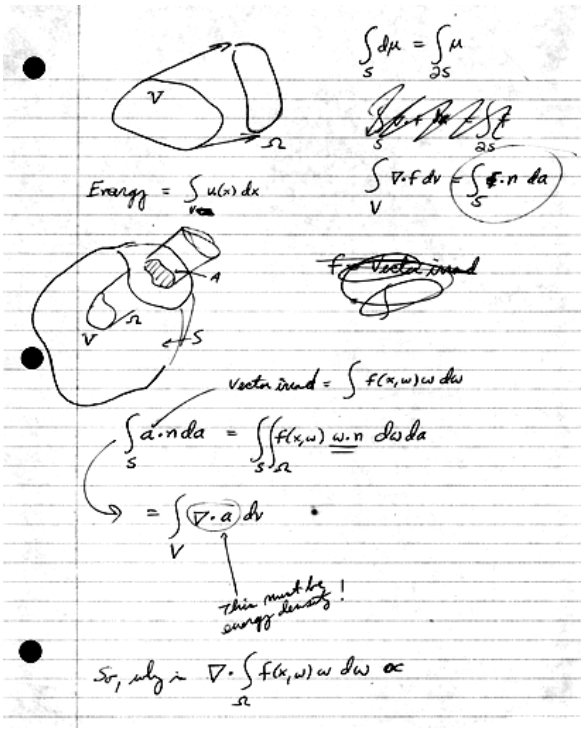

Figure 1: *A page of scribbled ideas consisting of sketches, fragments of mathematics, and annotations, from one of the author's numerous dusty notebooks.*

There appear to be only two ways to improve the situation: 1) impart greater discipline in human problem solving, or 2) imbue machines with a greater tolerance for the haphazard processes by which humans tend to solve problems. Of these, only the latter strategy offers much hope of long-term success. While nature has endowed us with remarkable abilities to think and to plan, our innate reasoning is frequently at odds with sound logic [13]. While the human brain is superbly adapted for complex social interactions and coping with a hostile environment, its heuristics are less trustworthy in other contexts. In particular, we are far more adept at quick intuitive assessments, particularly in social contexts, than complex inferences in formal logic [6]. Furthermore, myriad uncon-

scious processes generate insights and shift our awareness, contributing to the apparent randomness of human problem solving. It is clearly unrealistic to expect this to change.

On the other hand, the task of making machines more human-like in their behavior is an immense undertaking, as evidenced by the scant progress toward this goal during the past thirty years. Fortunately, a machine needn't pass the Turing test in order to meaningfully support human creativity. We shall argue that one step in the right direction is to apply non-monotonic reasoning as well as its counterpart, *belief revision* [12, 16], to human-computer interaction, making computers more resilient to the types of errors humans make. In essence, the dialog between human and computer should be embedded in a more accommodating logic; one in which the beliefs of both agents are treated as *tentative*, and therefore subject to revision in the light of new information. This allows for liberal use of default assumptions, which in turn help to smooth over ambiguities that arise, especially during problem formulation.

A second step toward more effective machine-assisted problem solving is to incorporate diagrammatic interaction [2, 5]. Although mechanical recognition of hand-drawn figures is a tremendously challenging problem, there are numerous domains in which the highly constrained syntax of the diagrams vastly simplifies the task of recognition: examples include state diagrams for finite automata and pictorial representations of sets.

Once a problem is formulated, how can a computer assist in the discovery of a solution? Certainly Ashby's notion of a machine that produces plausible solutions and then helps to sift through them is not far from the mark. However, production of possible solutions cannot be done blindly, nor can the search be exhaustive. While one could generate random strings of symbols in search of a proof or refutation of some conjecture, such an approach would be highly unlikely to turn up many new discoveries. To illustrate several plausible techniques, we outline a hypothetical proof scenario.

3 Anatomy of a Proof

We now consider a simple mathematical problem from linear algebra, and suggest how a computer might assist in its solution. Although the problem is elementary, it will nevertheless serve to illustrate some basic points. The problem is as follows: Given a monic polynomial of degree n,

$$p_n(x) \;=\; a_0 + a_1 x + \cdots + a_{n-1}x^{n-1} + x^n, \qquad (1)$$

construct a matrix whose eigenvalues are the roots of this polynomial. In other words, find an $n \times n$ matrix $\mathbf{M}_n$ whose *characteristic polynomial* is p_n up to a constant factor:

$$\det\left(\mathbf{M}_n - \lambda\mathbf{I}\right) \;=\; k\,p_n(\lambda), \qquad (2)$$

where k is a non-zero constant. That such a matrix exists can be seen immediately, since the diagonal matrix

$$\begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{bmatrix} \qquad (3)$$

clearly satisfies Equation (2), where $\lambda_1, \ldots, \lambda_n$ are the (possibly complex) roots of p_n. However, the more interesting question is whether such a matrix can be constructed without first finding the roots of p_n. If this could be done directly from the coefficients $a_0, \ldots, a_{n-1}$ using a finite number of arithmetic operations, it would demonstrate that the eigenvalue problem subsumes polynomial root finding, and consequently cannot be solved algebraically in general [30].

This problem has a well-known solution, which is known as the *companion matrix* of the polynomial p_n [14]. For the purpose of illustration, we shall feign ignorance of this result and set off to discover our own solution, documenting some of the wrong turns and blind alleys.

As a routine starting point, we observe that the problem has a trivial solution when $n = 1$. The matrix $[-a_0]$ has the characteristic polynomial $-a_0 - \lambda$, which is -1 times the monic polynomial $p_1(\lambda) = a_0 + \lambda$. So far so good. Next, we might consider the matrix

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix},$$

with the characteristic polynomial $\lambda^2 - (a+d)\lambda - bc$. We observe that any 2nd-degree monic polynomial can be obtained with a judicious choice of a, b, c, and d. The difficulty, of course, is that there are many such choices, and it is not obvious which choice, if any, will generalize to $n > 2$. Unfortunately, the problem already begins to get a bit messy with $n = 3$. Where do we go from here?

It's time to try induction. Suppose we can construct the matrix $\mathbf{M}_n$ from any given p_n; can we then construct $\mathbf{M}_{n+1}$ from p_{n+1}? There seem to be many possible ways to proceed. Should we represent the $(n+1)$'st degree monic polynomial as

$$p_{n+1}(x) \;=\; \alpha\,p_n(x) + x^{n+1} \qquad (4)$$

for some constant α, or as

$$p_{n+1}(x) \;=\; \alpha + x\,p_n(x) \qquad (5)$$

for a different choice of α? Should we create the larger matrix $\mathbf{M}_{n+1}$ by modifying the matrix $\mathbf{M}_n$ in some way, or simply by padding,

$$\mathbf{M}_{n+1} = \begin{bmatrix} * & * & \cdots & * \\ * & & & \\ \vdots & & \mathbf{M}_n & \\ * & & & \end{bmatrix}, \tag{6}$$

and if so, how do we fill in the new entries? Here's where we could use a bit of luck. If we were to guess the correct structure, it would likely be a straightforward matter to prove it by induction (and thereby cover our tracks).

For the sake of brevity, let us suppose that we have (correctly) chosen to pursue equations (5) and (6). In reality, of course, several unsuccessful attempts might have been made before discovering this combination. Expressing $\mathbf{M}_{n+1}$ as

$$\mathbf{M}_{n+1} = \begin{bmatrix} \beta_n & \mathbf{u}_n^{\mathrm{T}} \\ \mathbf{v}_n & \mathbf{M}_n \end{bmatrix}, \tag{7}$$

where $\beta_n \in \mathbb{R}$ and $\mathbf{u}, \mathbf{v} \in \mathbb{R}^n$, we are left with the task of determining how β_n, $\mathbf{u}_n$, and $\mathbf{v}_n$ should be constructed. Clearly, there must be some connection to each new coefficient added to the polynomial, so we try

$$\begin{aligned} \beta_n &= \beta_n(\alpha), \\ \mathbf{u}_n &= \mathbf{u}_n(\alpha), \\ \mathbf{v}_n &= \mathbf{v}_n(\alpha), \end{aligned}$$

where α is the constant appearing in Equation (5). Now, when $n = 1$, we have $\mathbf{M}_1 = [-a_0]$, from which it follows that

$$\begin{aligned} \beta_1 &= 0, \\ \mathbf{u}_1\mathbf{v}_1 &= -\alpha, \end{aligned} \tag{8}$$

where $\mathbf{u}_1$ and $\mathbf{v}_1$ are scalars corresponding to the off-diagonal entries of the 2×2 matrix. If we surmise that $\beta_n = 0$ for all n, we are then left with several other choices in generalizing to arbitrary $n > 2$. How should $-\alpha$ be "factored" into a product of two numbers, as required by Equation (8), and how should the scalars $\mathbf{u}_1$ and $\mathbf{v}_1$ be generalized to vectors $\mathbf{u}_n$ and $\mathbf{v}_n$ when $n > 1$? Furthermore, do these strategies depend on n? We could attempt the answer these questions through a deeper analysis of the problem, or we could simply guess, perhaps using past experience with similar problems, esthetic considerations, or even blind luck. In any case, a guess can be discarded immediately if it fails on a simple example, and ultimately proven symbolically if indeed it is correct.

Listed below are some of the obvious guesses that one might try for "factoring" a real number:

$$\begin{aligned} \phi_1(x) &= (1, x), \\ \phi_2(x) &= (x, 1), \\ \phi_3(x) &= (-x, -1), \\ &\vdots \end{aligned}$$

Here each type of guess has been encoded as a function from $\mathbb{R}$ to $\mathbb{R}^2$, although one would typically not employ this degree of formality in searching for a solution.

Once we have chosen a method ϕ_i to produce the two factors, they must be mapped to two vectors of the appropriate size, and in such a way that they reduce to the identity mapping when $n = 1$. Here are some of the more obvious possibilities for such a mapping:

$$\begin{aligned} \nu_1^n(x) &= (x, 0, \ldots, 0)^{\mathrm{T}}, \\ \nu_2^n(x) &= (0, \ldots, 0, x)^{\mathrm{T}}, \\ \nu_3^n(x) &= (x, x, \ldots, x)^{\mathrm{T}}, \\ &\vdots \end{aligned}$$

where all of the vectors on the right are in $\mathbb{R}^n$. By selecting the right functions, and composing them in the right way (with the aid of the projection function π_i, which selects the i'th coordinate of an n-tuple), two correct solutions emerge. Specifically, setting

$$\begin{aligned} \mathbf{u}_n(\alpha) &= \nu_2^n(\,\pi_1\phi_2(-\alpha)\,), \\ \mathbf{v}_n(\alpha) &= \nu_1^n(\,\pi_2\phi_2(-\alpha)\,) \end{aligned}$$

and then recursively applying the construction depicted in Equation (7) down to the base case $\mathbf{M}_1$, we generate the matrix

$$\mathbf{M}_n = \begin{bmatrix} 0 & 0 & \ldots & 0 & -a_0 \\ 1 & & & & -a_1 \\ & 1 & & & -a_2 \\ & & \ddots & & \vdots \\ & & & 1 & -a_{n-1} \end{bmatrix},$$

which is precisely the form of the companion matrix given by Golub and Van Loan [14]. Similarly, the combination

$$\begin{aligned} \mathbf{u}_n(\alpha) &= \nu_1^n(\,\pi_1\phi_1(-\alpha)\,), \\ \mathbf{v}_n(\alpha) &= \nu_2^n(\,\pi_2\phi_1(-\alpha)\,), \end{aligned}$$

generates the transpose of the above matrix, which has the same characteristic polynomial. Two further variants arise when the matrix $\mathbf{M}_n$ appears as the upper-left block in Equation (6).

3.1 Opportunities for Mechanized Assistance

The problem described above could obviously be worked out entirely on paper, with no mechanical assistance whatsoever. On the other hand, the construction might also have been discovered purely mechanically. Since any problem of mathematics is conceivably solvable by either means, the question of interest here is whether there exists a middle ground in which there is a synergy between human and machine. If so, how should the work be partitioned, and how should information be exchanged between the two?

The problem of the companion matrix suggests several ways in which this synergy may be realized. First, it suggests the utility of automatically testing the feasibility of guesses on simple test cases, either symbolically or numerically. This type of testing can easily be the most onerous aspect of solving a problem, yet it is invaluable for quickly ruling out approaches that are obviously wrong. In the author's prototype system, for example, the (re)discovery of the companion matrix was partially automated using symbolic computation of the characteristic polynomials followed by numerical evaluation for testing all compositions of the guesses described earlier. After constructing an arbitrary p_n and the corresponding matrix $\mathbf{M}_n$, for some $n > 2$, the ratio

$$\frac{p_n(x)}{\det\left(\mathbf{M}_n - x\mathbf{I}\right)} \tag{9}$$

was evaluated at n distinct points and checked for constancy. In this way polynomials that differed by more than a constant factor were quickly detected, using a small number of evaluations on average. This test is a valuable precursor to pure symbolic manipulation of polynomials, which is typically much more costly.

Another area in which a machine can assist is in the selection of the heuristics as embodied by the functions $\phi_1, \phi_2, \ldots$ and $\nu_1^n, \nu_2^n, \ldots$ introduced above. While there is no guarantee that a simple trick used in one context will be helpful in another, it is not uncommon for solution techniques to extend beyond their intended domain. Coupled with a capability for quickly testing plausible guesses on specific problem instances, automatic application of a large collection of heuristics becomes a valuable tool for discovery.

Naturally, when a heuristic search such as the one described above does not succeed, it may indicate that the construction is impossible, or simply that the appropriate strategy has not yet been found. When all existing heuristics have been exhausted, the machine can still assist by discerning patterns of failure (via still other heuristics) and by indicating opportunities for new heuristics to the user.

Figure 2: *Illustration depicting a key step in the proof of the Hahn-Banach theorem. Reproduced with kind permission of C. M. Strauss.*

4 The Role of Diagrams

Diagrams play many roles in problem solving. In form they range from indecipherable doodles to complete representations with precisely defined semantics. It is suggested here that there is a vast and interesting middle ground occupied by diagrams that provide hints and serve as signposts for formal structure.

An amusing anecdote that nicely illustrates one extreme in the spectrum is related in a paper by C. M. Strauss [28]. Therein is the story of a professor who, upon reaching an impasse in the proof of the Hahn-Banach theorem during a lecture, drew the illustration shown in Figure 2, and immediately saw the correct way to proceed. While one may speculate that the illustration somehow evoked the concept of "separation," clearly the diagram played no formal role in the proof and conveyed essentially no information; the semantic content was entirely in the mind of the professor.

At the other end of the spectrum are diagrams with complete semantics, such as those that are isomorphic to mathematical constructs. For instance, the diagrams in Figure 3 denote sets A, B, and C, and several relations that hold among them. The Venn diagram on the left depicts the properties $A \cap C \subseteq B$ and $(B - A) \neq \emptyset$. The Euler diagram on the right depicts the relations $B \subseteq A$ and $B \cap C = \emptyset$. Both types of diagram have a precise syntax and meaning [15]. As another example, consider Figure 4, which depicts a Turing machine that increments a binary-encoded number. This state diagram is isomorphic to the formal definition of the machine, and is thus complete and unambiguous. Finite state machines and push-down automata admit similar diagrammatic representations. Such diagrams are far easier to comprehend than their set-theoretic counterparts, yet sacrifice nothing in terms of semantics.

Diagrams can also carry meaning without being fully isomorphic to the concepts that they depict. Such diagrams can be used to clarify or suggest steps in a rigorous proof, as is most evident in the case of geometry. However, even topological concepts can be conveyed through diagrammatic representations. Let us consider a simple

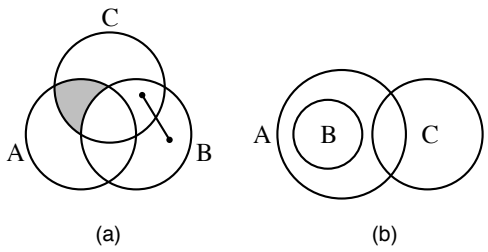

Figure 3: *Diagrams with well-defined formal semantics: (a) Venn diagrams, and (b) Euler diagrams.*

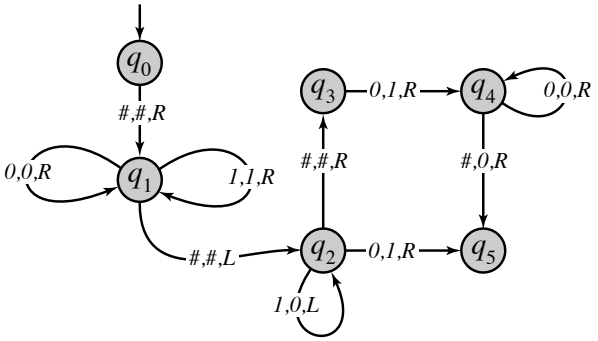

Figure 4: *A simple Turing machine, completely described in terms of its state diagram.*

theorem from real analysis that rests upon purely metric and topological notions and explore a hypothetical proof procedure that is guided by highly abstract diagrams.

Suppose that we wish to prove the following statement: a set S is *relatively compact* if it is *totally bounded*. By a relatively compact set we mean a set in which every infinite sequence contains a Cauchy sub-sequence. By a total bounded set we mean a set that can be "covered" by a finite number of open ϵ*-balls* (essentially disks of radius ϵ) for any $\epsilon > 0$. Both concepts are meaningful in any metric topology, and technically involve no geometry at all.

To prove the assertion we must somehow use total boundedness to extract a Cauchy sequence from an arbitrary infinite sequence in S. This can be accomplished with an iterative process that has a simple visual metaphor. Figure 5 shows a sequence of illustrations reminiscent of what one might draw in attempting to explain or discover the necessary connection.

In the figure, S depicts a totally bounded but otherwise arbitrary set, and S_0 is an infinite sequence in S; the second illustration shows a finite collection of ϵ-balls that cover S_0. The balls are depicted as circles, although the actual shapes are irrelevant. Since S_0, as a set, is also totally bounded by virtue of being in S, we can choose such a cover that is finite. The key insight is that the infinite sequence S_0 must then hit at least one of the ϵ-balls "infinitely often," which is a common idiom of analysis. In the illustration, each ϵ-ball containing infinitely many elements of S_0 is emphasized. We denote by S_1 the subsequence of S_0 contained in one such ball, and "zoom in"; it becomes evident that this process can be repeated indefinitely, since each sub-sequence retains the essential properties of the original sequence. Finally, by choosing a single point from each of $S_1, S_2, S_3, \ldots$, we produce the desired Cauchy sub-sequence of S_0, as the tail of this sequence is contained within ever shrinking neighborhoods.

This example illustrates the potential for diagrammatic representations in exploring topological concepts. The role of the diagram is to serve as a guide for the mathematics, not as a substitute for it. Once the underlying principles are discovered, a rigorous symbolic proof can be constructed.

Such idiomatic illustrations lend themselves to random construction by computer. These representations can be used to access and apply relevant concepts, such as finite covering sets. To help the user avoid incorrect assumptions, the "arbitrary" examples that are generated should include reminders of valid special cases, such as a single point appearing infinitely many times in a sequence, or multiple balls in a cover being hit infinitely often. Diagrams of this nature can enhance computer-assisted discovery by associating concepts with appropriate imagery, and providing visual cues for salient properties.

5 Implications for Human-Computer Interaction

To some extent it is possible to mimic mathematical reasoning on a computer [7]. When a human participates in the solution process, however, many special considerations involving the interface arise that have no counterpart in purely mechanical processes. We now identify some of these considerations, and discuss how they influence human-computer interaction.

In principle one can perform any manner of experiment using conventional computer algebra systems and theorem provers, and thereby synthesize proofs or make discoveries. In practice, however, experimentation is hampered by representations that are somewhat difficult to specify and to manipulate. While a conjecture may be succinctly expressed with a diagram or concise mathematical notation, transcribing it into the language of the system and constructing or verifying basic test cases can be tedious. To the extent that these obstacles are lessened, and promising avenues made more perspicuous, the likelihood of finding useful connections increases; that is, we encourage serendipity.

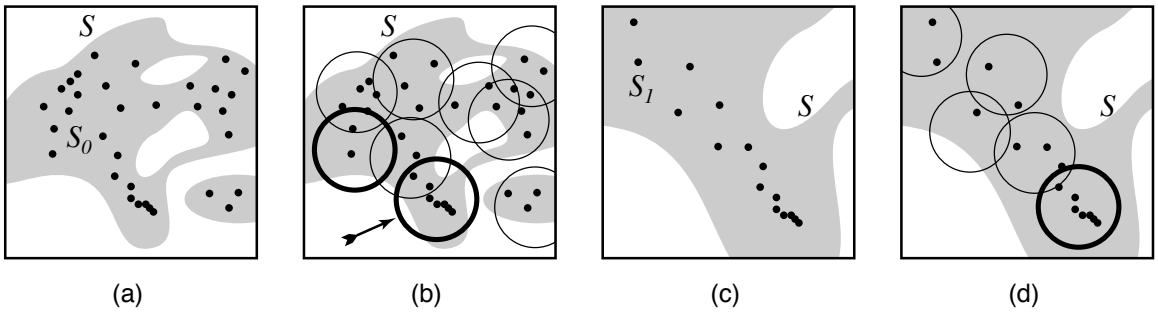

Figure 5: *Steps in a topology proof guided by diagrams that serve an informal role. (a) Representation of an infinite sequence S_0 within a totally bounded set S. (b) Representation of a finite covering of S_0 by ϵ-balls. Balls that meet S_0 infinitely often are emphasized. (c) Closeup of the subset S_1 within the selected ϵ-ball. (d) An ϵ-covering of S_1, but with a smaller ϵ. The process can be repeated indefinitely.*

We shall consider some of the difficulties encountered with hand-written notation, both because pen input can be very expressive, and because it illustrates a wide variety of errors and ambiguities that must be dealt with. Pen-based computer interfaces were first developed as a tool for geometric construction and manipulation [29], and shortly thereafter found application in symbolic mathematics [3]. The processing of hand-drawn mathematical notation continues to be a challenging research problem [19, 27] as it requires both character recognition and two-dimensional parsing. Part of its appeal is that it offers the opportunity to combine symbolic, diagrammatic, and gestural information [9, 24].

Figure 6 shows a hand-written equation that presents a number of challenges for a hypothetical pen-based mathematics system. First, since the symbols "$2a$" are not very legible, they are mistakenly recognized as the symbol "u," which in turn makes the ellipsis ambiguous. On the second line, the upper limit of the product does not agree with the factor $\cos(2^k a)$ on the line above; it is likely that either the k or the $k-1$ is not what was intended. The symbols "$\sin 2^k a$" are initially interpreted as "$\sin(2^k)a$," but "$\sin(2^k a)$" was intended. Finally, the denominator is initially interpreted as "$2^k \sin d$" rather than "$2^k \sin a$." Correctly interpreting this expression is challenging for humans and machines alike.

Fortunately, problems of this nature can often be rectified if the decisions made during interpretation are re-examined and possibly revised when additional information becomes available or when inconsistencies arise. For example, the ambiguity of the ellipsis indicates that something is awry, so recognition and stroke grouping can be re-examined to identify any "nearby" interpretations that might disambiguate it. The low recognition confidence of the "u" points to it as the likely culprit, and re-grouping the strokes reveals the correct interpretation; this process depends on a scheme similar to that described by Smithies et al. [27].

The misrecognition of the "a" as a "d" and the grouping of "$2^k a$" are more interesting, as the equation might actually be correct as it stands. However, two factors invite other interpretations to be considered. First, the equality of the last two expressions would be directly testable if not for the free variable d, and second, "a" is the next-most-likely interpretation of the symbol "d." Numerical verification of the identity

$$\prod_{j=0}^{k-1} \cos\left(2^j a\right) = \frac{\sin\left(2^k a\right)}{2^k \sin a} \tag{10}$$

then provides concrete evidence that this is indeed what was intended. The corresponding changes are adopted until other evidence prompts the system to revisit these decisions yet again.

As an example that illustrates the use of higher-level context in resolving conflicting information, consider the hand-written equation shown in Figure 7. Appearing on the right hand side is a diagrammatic representation of a matrix, which we shall denote by **U**. Several default assumptions can be immediately made about this equation, including

1. **M** is an arbitrary matrix.
2. **Q** is a non-singular matrix.
3. **U** is upper-triangular.

As a consequence of these assumptions, it also follows that all the matrices are square. In isolation, these default assumptions are consistent. However, now suppose that the problem to be solved is to construct **Q** from **M** so that the equality holds. By default, it is assumed that such a construction is to be *direct*, or *closed-form*, and this leads

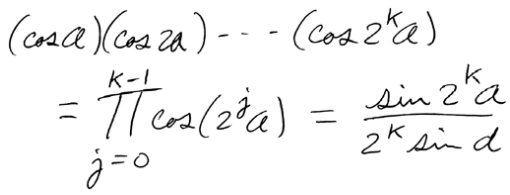

Figure 6: *A hand-written equation with several errors and ambiguities*.

to a contradiction. Specifically, the assumptions taken together imply the existence of a closed-form solution to the general eigenvalue problem; the companion matrix was instrumental in proving this impossible. Thus, it is necessary to retract (or weaken) some prior belief about the symbols. For example

1. **M** has some special form.
2. **Q** is computed iteratively, not directly.
3. **U** is upper Hessenberg, not upper triangular.

Any of these choices would remedy the problem, yet there is no purely logical means to choose among them. Here we rely upon the relative likelihoods of the default assumptions to break the tie. Most suspect is the structure of the matrix **U**, as it has been inferred from a crude diagram. Thus, we retract the assumption and replace it with the next-most-likely interpretation, which is that of an *upper Hessenberg* matrix; that is, a matrix that is upper triangular except for entries along the sub-diagonal. This choice is further supported by the fact that the two representations can be easily confused. We leave aside the difficult problem of how this new interpretation is communicated to or verified by the user; a continuum of behaviors may be appropriate, depending on the needs and preferences of the user. We instead focus on tools that can assist in identifying and resolving ambiguities and contradictions such as these.

6 Non-monotonic Reasoning

When a contradiction arises, the *belief set* maintained by the mathematics system must be repaired or it becomes useless. Consequently, as new information is acquired, it may result in previous assumptions being rejected or modified. The reasoning embodied by such a system is thus referred to as *non-monotonic*. Recent work in *belief revision* provides some insight into the non-monotonic nature of cooperating agents [12]; in fact belief revision is essentially a formalization of non-monotonic reasoning in which *rationality axioms* constrain how an agent responds to conflicting information.

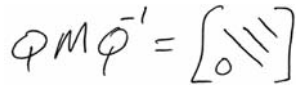

Figure 7: *A hand-printed similarity transformation in which the structure of one matrix is specified diagrammatically.*

6.1 Belief Revision

First, let us assume that the "beliefs" of an agent are encoded as a collection of logical formulas, or *sentences*, which are formed using basic logical connectives, propositional variables, and perhaps quantifiers. Let $\mathcal{B}$ denote a collection of beliefs defined by some finite *base* of propositional formulas (essentially *axioms*), and all sentences that logically follow from these formulas according to a specified set of inference rules. That is, we assume that $\mathcal{B}$ is *closed* under logical inference, or *entailment*; for example, if $x \in \mathcal{B}$ and $x \to y \in \mathcal{B}$, then $y \in \mathcal{B}$, by modus ponens.

A belief set can be *expanded* by adding a sentence to the base, and implicitly including all new inferences. Thus, simple *expansion* is defined by

$$\mathcal{B} + x \;=\; \{y \mid \mathcal{B} \cup \{x\} \vdash y\}\,,$$

where the sentence x may or may not be consistent with the beliefs already in $\mathcal{B}$. If $\neg x \in \mathcal{B}$, then $\mathcal{B} + x$ contains all possible sentences, since it entails a contradiction. Consequently, this method of accumulating information is useless when conflicting facts arise.

To accommodate potential contradictions, we must adopt the use of a *revision operator*, denoted by $\overset{\circ}{+}$, which ensures that the revised belief set $\mathcal{B} \overset{\circ}{+} x$ remains consistent provided that $\mathcal{B}$ was consistent to begin with, and x is not itself a contradiction [12].

Revision operators have several fundamental properties. First, they are not uniquely determined by logical considerations alone; that is, for any given belief set $\mathcal{B}$ and sentence x, there may be numerous logically consistent ways to retract information from $\mathcal{B}$ so that x can be safely added. Second, under mild assumptions revision is completely determined by *contraction*; that is, by the strategy for *removing* information from a belief set. This second fact follows from the *Levi identity*,

$$\mathcal{B} \overset{\circ}{+} x = \left(\mathcal{B} \overset{\circ}{-} \neg x\right) + x, \tag{11}$$

where $\overset{\circ}{-}$ denotes the contraction operator. Equation (11) holds under the rationality axioms proposed by Alchourrón, Gärdenfors, and Makinson [1], which state, for example, that $x \in \mathcal{B} \overset{\circ}{+} x \subseteq \mathcal{B} + x$, and that $\mathcal{B} \overset{\circ}{+} x = \mathcal{B} + x$ when x is consistent with $\mathcal{B}$. Furthermore, these axioms

require the revision operator to be sensitive only to meaning and not syntax. Thus, when x and y are logically equivalent, it must follow that $\mathcal{B} \overset{\circ}{+} x = \mathcal{B} \overset{\circ}{+} y$.

By imposing these rationality axioms, the possible interpretations of contraction and revision are constrained but not entirely determined; thus, as we saw in the previous examples, we require some extra-logical information to decide which revision is most justified. One such source of additional information is *epistemic entrenchment* [11], which is a partial ordering on the beliefs that indicates our degree of commitment to them, or our confidence in them. Alchourron *et al*. [1] have shown how to characterize the possible revisions that satisfy the rationality axioms, which precisely demarcates the role of extra-logical heuristics such as entrenchment and confidence measures. This is a promising theoretical foundation for constructing agents that can reason non-monotonically, and hence cope with some degree of ambiguity.

7 Autonomous Assistants

We now summarize some of the lessons embodied in the examples discussed, and translate them into desirable properties of a mathematical assistant. First, and most fundamentally, the user of such a system should not be forced into a rigidly formal development. Requiring that each step be thoroughly specified is overly constraining and is antithetical to the way people tend to work. People are quite accustomed to carrying on in the face of uncertainty, so it is desirable, if not expected, that an assistant will have this characteristic also. The use of hand-written information and diagrams are consistent with this mandate, although not a fundamental requirement.

An immediate consequence of this freedom is that the assistant must be capable of applying tentative default assumptions about the meanings of symbols as well as the objectives of the user. The tentativeness of the assumptions leads to non-monotonic reasoning on the part of the assistant, which can be characterized in terms of its strategy for belief revision.

Since contradictions might be revealed only after significant computation, either through logical inference (i.e. theorem proving) or numerical experiments, changes in the assistant's beliefs may occur at seemingly random times. It is therefore reasonable to require that the human-computer dialog never be suspended pending the outcome these computations; thus they must be relegated to independent threads of execution. Since neither the human nor the assistant need ever be completely stalled due to information required from the other, their activities are largely decoupled. The picture that emerges is that of an *autonomous assistant*.

Multi-threaded execution can also be useful in carrying out numerical experiments and symbolic computations simultaneously, since either path may turn up useful information, or fail to do so even after significant computation [32]. In addition to the use of concurrency [10], default assumptions, non-monotonic reasoning, and heuristics that encapsulate previous tricks and *plausible reasoning* [22], there are many other ways in which the usefulness of an autonomous assistant may be enhanced. For example, it would be useful to devise ways in which to inform the user of pending computations and current beliefs of the assistant. This will likely require the assistant to periodically inject unsolicited information, and may therefore benefit from assistants that can mimic additional human attributes, such as "politeness" or "assertiveness."

Acknowledgments

The author wishes to thank Kevin Novins, Boris Dimitrov, Al Barr, and Erin Shaw for stimulating conversations and valuable feedback. This work was supported in part by the NSF Science and Technology Center for Computer Graphics and Scientific Visualization, the Army Research Office Young Investigator Program (DAAH04-96-100077), and the Alfred P. Sloan Foundation.

8 References

[1] Carlos E. Alchourrón, Peter Gärdenfors, and David Makinson. On the logic of theory change: Partial meet contraction and revision functions. *Journal of Symbolic Logic*, 50(2):510–530, June 1985.

[2] Michael Anderson and Robert McCartney. Diagrammatic reasoning and cases. In *Proceedings of the 13'th National Conference on Artificial Intelligence (AAAI-96)*, pages 1004–1009, Portland, Oregon, August 1996.

[3] Robert H. Anderson. *Syntax-Directed Recognition of Hand-Printed Two-Dimensional Mathematics*. PhD thesis, Harvard University, Cambridge, Massachusetts, 1968.

[4] W. Ross Ashby. Design for an intelligence-amplifier. In C. E. Shannon and J. McCarthy, editors, *Automata Studies*, pages 215–234. Princeton University Press, Princeton, New Jersey, 1956.

[5] Jon Barwise and John Etchemendy. Heterogeneous logic. In G. Allwein and J. Barwise, editors, *Logical Reasoning with Diagrams*, chapter 8, pages 179–200. Oxford University Press, New York, 1996.

[6] Martin D. S. Braine. On the relation between the natural logic of reasoning and standard logic. *Psychological Review*, 85(1):1–21, January 1978.

[7] Alan Bundy. *The Computer Modelling of Mathematical Reasoning*. Academic Press, New York, 1983.

[8] Chin-Liang Chang and Richard Char-Tung Lee. *Symbolic Logic and Mechanical Theorem Proving*. Academic Press, New York, 1973.

[9] James D. Foley and Victor L. Wallace. The art of natural graphic man-machine conversation. *Proceedings of the IEEE*, April 1974.

[10] Emden R. Gansner and John H. Reppy. A multi-threaded higher-order user interface toolkit. In Bass and Dewan, editors, *User Interface Software*, chapter 4. John Wiley & Sons, New York, 1993.

[11] Peter Gärdenfors. Epistemic importance and minimal changes of belief. *Australian Journal of Philosophy*, 62(2):136–157, June 1984.

[12] Peter Gärdenfors. Belief revision: An introduction. In Peter Gärdenfors, editor, *Belief Revision*, volume 29 of *Cambridge Tracts in Theoretical Computer Science*, pages 1–28. Cambridge University Press, New York, 1992.

[13] Thomas Gilovich. *How We Know What Isn't So: The Fallibility of Human Reason in Everyday Life*. Free Press, New York, 1991.

[14] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. Johns Hopkins, Baltimore, 1983.

[15] Eric M. Hammer. *Logic and Visual Information*. CSLI Publications, Stanford, California, 1995.

[16] Hirofumi Katsuno and Alberto O. Mendelzon. On the difference between updating a knowledge base and revising it. In Peter Gärdenfors, editor, *Belief Revision*, volume 29 of *Cambridge Tracts in Theoretical Computer Science*, pages 183–203. Cambridge University Press, New York, 1992.

[17] Arthur Kornberg. Of serendipity and science. *Stanford Medicine Magazine*, 12(4), Summer 1995.

[18] Douglas Bruce Lenat. *AM: An Artificial Intelligence Approach to Discovery in Mathematics as Heuristic Search*. PhD thesis, Stanford University, July 1976.

[19] Erik G. Miller and Paul A. Viola. Ambiguity and constraint in mathematical expression recognition. In *Proceedings of the 15'th National Conference on Artificial Intelligence (AAAI-98)*, pages 784–791, Madison, Wisconsin, July 1998.

[20] Tsuyoshi Murata, Masami Mizutani, and Masamichi Shimura. A discovery system for trigonometric functions. In *Proceedings of the 12'th National Conference on Artificial Intelligence (AAAI-94)*, pages 645–650, Seattle, Washington, July 1994.

[21] Marko Petkovšek, Herbert S. Wilf, and Doron Zeilberger. *A = B*. AK Peters, Ltd., Wellesley, Massachusetts, 1996.

[22] G. Polya. *Mathematics and Plausible Reasoning*, volume II: *Patterns of Plausible Inference*. Princeton University Press, New Jersey, 1954.

[23] Royson M. Roberts. *Serendipity: Accidental Discoveries in Science*. John Wiley & Sons, New York, 1989.

[24] Dean Rubine. Specifying gestures by example. *Computer Graphics*, 25(4):329–337, July 1991.

[25] Alan H. Schoenfeld. *Mathematical Problem Solving*. Academic Press, New York, 1985.

[26] N. Shankar. *Metamathematics, Machines, and Gödel's Proof*, volume 38 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, New York, 1994.

[27] Steve Smithies, Kevin Novins, and James Arvo. A handwriting-based equation editor. In *Proceedings of Graphics Interface '99*, Kingston, Ontario, June 1999.

[28] Charles M. Strauss. Computer-encouraged serendipity in pure mathematics. *Proceedings of the IEEE*, 62(4):493–495, April 1974.

[29] Ivan E. Sutherland. *SKETCHPAD: A Man-Machine Graphical Communication System*. PhD thesis, Massachusetts Institute of Technology, January 1963.

[30] Lloyd N. Trefethen and David Bau, III. *Numerical Linear Algebra*. Society for Industrial and Applied Mathematics, Philadelphia, 1997.

[31] Jan L. A. van de Snepscheut. Mechanized support for stepwise refinement. Technical Report 187, California Institute of Technology, Pasadena, California, January 1994.

[32] Richard S. Varga. *Scientific Computation on Mathematical Problems and Conjectures*. Society for Industrial and Applied Mathematics, Philadelphia, 1990.

[33] Stephen Wolfram. *Mathematica: A System for Doing Mathematics by Computer*. Addison-Wesley, Reading, Massachusetts, second edition, 1993.

[34] Kenneth M. Yip. *KAM: A System for Intelligently Guiding Numerical Experimentation by Computer*. MIT Press, Cambridge, Massachusetts, 1991.