What AI Practitioners Should Know about the Law

Part One

Steven J. Frank

This is Part 1 of a two-part article. Part 2 covers tort liability and computers as expert witnesses. It will appear in the Summer 1988 issue of AI Magazine. Technological developments that remove ever-increasing numbers of cognitive tasks from human control will alter the assumptions on which current legal rules are based. These rules will have a growing impact on AI researchers and entrepreneurs as their work reaches a growing audience of beneficiaries. In order to accommodate the needs of practitioners and their recipients, courts and lawmakers will be forced to reevaluate principles whose foundations were developed well before the implications of advanced technology could have been predicted. This article attempts to identify areas of law in which the need for accommodation will be greatest and provide some insight into the process and the direction of change.

Technological advances are usually shadowed by changes in the legal system. As the daily subject matter facing judges and lawyers evolves, new issues, priorities, and dangers emerge. The dimensions of familiar rights and obligations soon become outmoded or appear increasingly ambiguous in the face of the unfamiliar, and some sort of response eventually ensues. Ideally, the response represents a process of accommodation between the old law and new needs. The reality, however, is often quite different.

In the United States, earlier decisions constrain the development of new legal principles by the judiciary. Lower-court judges are obligated to follow the past edicts of courts positioned above them in the judicial hierarchy, and even those standing at the highest levels are reluctant to disturb established precedent unless a departure is clearly warranted. Legislatures, although not bound by prior law, must strike a balance among competing political interests before enacting change. The point is that unless they are faced with a disturbance of cataclysmic proportions, the engines of legal progress will not likely move with breakneck speed.

One would imagine that such inertial propensities run contrary to the desires of the innovators whose activities and livelihoods are actually touched by regulation. One might envision those gallant entrepreneurial warriors, whose quest for the future has been interrupted by legal constraints of the past, struggling to extract movement from ponderous lawmakers. Yet, all too often, even the ones who stand to lose most from flawed legal precepts remain equally passive at the stage of policy formulation. For the scientist with a valuable new product, the existence of relevant law is frequently of interest only to the extent that it might get in the way. The process of change is viewed as an obstacle rather than an opportunity. Although common, this view is quite ironic. If those responsible for making the rules lack access to needed sources of expertise, those who must live with the rules will face needlessly imperfect handiwork.

As technological developments that remove ever-increasing numbers of cognitive tasks from human control alter the assumptions upon which current legal rules are based, the emergence of artificial intelligence (AI) systems will pose a significant challenge to the institutions responsible for creating law. It is important for innovators in the AI community to understand both the assumptions and the rules, so that the law can be improved where possible and avoided where necessary. As AI products continue to attain greater commercial prominence and, thereby, touch larger segments of society, AI practitioners will find three substantive legal fields increasingly influential: intellectual property, tort, and evidentiary law. This article is an attempt to introduce a changing legal structure to those who will be most directly affected by its evolution.

Intellectual Property

All software programs, regardless of purpose or complexity, exhibit a common vulnerability: They are expensive to create but relatively easy to reproduce. Over the years, software developers have relied on a triad of legal protection mechanisms to retain proprietary control over their work: copyright registration, patent protec-

tion, and the law of trade secrets. Each of these protective schemes emerged well before the advent of digital computers. As they have been recruited for application to contemporary technologies, deficiencies and inconsistencies have been discovered; pondered; and, to some extent, remedied. Yet despite recent efforts, the current system remains a patchwork scheme containing worrisome gaps and unnecessary overlap. The emergence of AI software can be expected to place additional strain on archaic conceptual foundations and overworked adaptations.

Copyright

Copyright registration was originally intended to provide authors with a means of protecting literary works.[1] As newer modes of aesthetic and intellectual expression became available, the terms of the Copyright Act were broadened to include them. In its present incarnation (the 1976 Act, Title 17 of the U.S. Code), copyright registration and protection are made available for "original work[s] of authorship fixed in any tangible medium of expression."[2] Computer software occupied a controversial status for many years. Although computer programs can be perceived and comprehended by humans, their chief function is not to inspire human thought. Programs are written to direct the operation of digital electronic components. This utilitarian role was considered by many to fall outside the spirit of a system intended to protect expressive creations.[3] Although written, a program is not a literary work in the everyday sense because its primary audience is nonhuman.

Despite the uncomfortable fit, by the early 1970s Congress became convinced that copyright protection for computer software was an idea whose time had come. Prior to enacting the 1976 Act, it created the National Commission on New Technology Uses of Copyrighted Works (CONTU) to consider the issue. The recommendations of CONTU were incorporated into the 1980 amendments to the 1976 Act. These provisions clarified the law in several respects, most notably by explicitly defining computer programs and bringing them under the terms of the Act. Registering software is now a simple matter of filing the proper form and submitting a nominal fee.[4] Far less clear, however, is exactly what is protected once such a filing is made.

Copyright Law: Ideas versus Expressions

A fundamental tenet of copyright law is that protection is limited to an author's expression; it does not extend to the underlying ideas.[5] This principle is well suited to forms of work whose valued content is wholly contained within the expression. For works of literature or films, protection of the author's chosen words, format, and style is sufficient to guard against theft of the creative effort. Computer programs, however, direct the execution of purposive tasks. Depending on the breadth and generality of the programmer's approach, many coding routes can be available to implement an identical set of symbolic or mathematic operations. Protecting a single version, then, ordinarily cannot protect the underlying solution.

Although this copyright restriction suggests that the level of protection will be enhanced by intrinsic programming limitations, the opposite is actually true. In fact, if a particular program captures its underlying methodology too well, copyright protection can be lost altogether: Courts will not support an infringement action if too few alternative means of expression exist. In such cases, the idea is said to merge with the expression, rendering both unprotectible.[6] This concept was applied to computer programs in Apple Computer, Inc. v. Franklin Computer Corp.,[7] which involved Apple's operating system software. As long as other programs can be written that perform the same function as the Apple software, said the court, that software can be copyrighted.[8]

The bottom line is that copyright protects implementation far better than function. It is most effectively used for software that embodies unoriginal ideas executed by means of a particular technique or in conjunction with a distinctive format. The simpler the programming task, the more reliable the protection is under the copyright laws.

Still, the subject matter of copyright must extend beyond bare expression in order for protection to be meaningful. If infringement actions could be won only against those who slavishly copy, registered works could be readily appropriated through trivial additions or deletions. The standard of comparison for infringement, therefore, has been defined by the courts as "substantial similarity" rather than pure congruence. It is through this deliberately inexact test that some ideas achieve copyright protection. Where expression ends and unprotectible ideas begin, however, is a line drawn differently by different judges.

The key inquiry lies in determining what it is that makes a particular computer program unique. A novel's distinctiveness does not reside solely in the author's choice of words or grammar but extends to the interplay of characters, plot, dialogue, and theme. Yet, no court would extend protection to all these elements because they inevitably embody ideas that would enlarge the author's monopoly to an unwanted extent. Were Shakespeare alive to copyright his works, his quill would not foreclose others from expounding on the death of Julius Caesar, nor would *West Side Story* infringe *Romeo and Juliet*. Instead, courts attempt to draw the line of substantial similarity by searching for a work's essential features, assessing the extent to which these features have been reproduced by the defendant and according protection to identifiable aspects of the work that fall short of the abstract. Textual similarity is obviously the easiest way to prove infringement, though it is not the only way. However, because literal copying is simpler to police objectively than paraphrasing or rearranging, plaintiffs are required to demonstrate more comprehensive duplication of nonliteral elements than of the words themselves.

For computer programs, similarity of source or object code must likewise comprise only a first step in deciding

... unless they are faced with a disturbance of cataclysmic proportions, the engines of legal progress will not likely move with breakneck speed.

whether two software packages are substantially similar. Program logic, data structures, and the form of output embody crucial elements of the programmer's design efforts. Courts must now apply established principles in unfamiliar territory. In a recent pronouncement on the subject, the United States Court of Appeals for the Third Circuit chose the following rule to distinguish idea from expression in the software context:

> [T]he line between idea and expression may be drawn with reference to the end sought to be achieved by the work in question. In other words, the purpose or function of a utilitarian work would be the work's idea, and everything that is not necessary to that purpose or function would be part of the expression of the idea.[9]

The court reasoned that where various means of achieving the desired goal exist, the particular route chosen is not essential to this goal—and, hence, can be protected as expression.

A similarly broad view was taken by a Georgia federal district court in Digital Communications Associates, Inc. v. Softklone Distributing Corp.[10] In this case, the court upheld the validity of Digital Communications Associates' (DCA) copyright of its CROSSTALK XVI status screen and found the status screen of its competitor's product to infringe this copyright. The "idea," said the court, was the process or manner by which the status screen operates and the "expression" the method of communication to the user. Applying this rule, the court found (1) the use of a screen to reflect the program's status, (2) the use of a command-driven program, and (3) the typing of two symbols to activate a specific command to be ideas. Portions of the status screen unrelated to program function, such as the arrangement and style of displayed terms, were viewed as protectible expression.

Current commercially targeted AI products, such as expert systems, derive their capabilities from ideas to a much greater degree than from program expression. Before actual programming can even begin, a working inference structure must be developed through laborious debriefing of human experts and thorough exploration of possible interactions among the distilled rules. Once the knowledge base is defined, however, computer implementation is straightforward; indeed, advertisements routinely appear in technical journals for expert system-development software designed to provide an empty shell into which completed knowledge structures can be loaded.

Such idea-driven software systems make poor candidates for copyright protection. Although the opinions in Whelan (see footnote 9) and DCA are relatively liberal, it is questionable whether they can be extended beyond program-bound features such as display formats, system architecture, and coding patterns. The knowledge base of a complex production system represents a working methodology for drawing conclusions within a specific domain, regardless of whether it is actually reduced to practice on a computer and, thus, can be said to possess an existence separate from its facilitative software. Whether courts will view such systems as lying closer to mathematical algorithms, which are purely cognitive and, therefore, not copyrightable, or the computational structures and organizations that have been accorded protection, remains to be seen.

Copyright Law: The Problem of Fixation

An essential prerequisite to obtain copyright is "fixation in a tangible medium of expression."[11] The price an author pays for protection under the copyright laws is the work's availability to the public. The Copyright Office requires that some form of the work be deposited as a condition of registration, ensuring access both for inspection and as a means of verifying authenticity in later infringement actions.

Fixation occurs when the work assumes a form sufficiently permanent or stable to permit it to be perceived, reproduced, or otherwise communicated for a period of more than transitory duration.[12] For the kinds of static compositions originally contemplated as objects of copyright, the

Questions of derivative ownership will arise as software becomes capable of producing integrated, reusable solutions to user-defined problems based on less and less input data.

requirement is fulfilled as soon as the work enters a form more tangible than the author's imagination. A problem is also not posed for current-day software: Although program code undergoes various transformations during program execution, the basic structure remains sufficiently stable that no one is terribly worried about the loss of copyright for intermediate states.

The development of computer programs capable of self-modification and self-update might well result in classes of software incapable of proper fixation. Nonmonotonic logic, the basis for much of the current research in machine learning, illustrates the kind of dynamics that can prevent the description of program contents in a manner acceptable to the Copyright Office. Program logic is said to be nonmonotonic when the introduction of new information results in the production of revised conclusions.[13] Although execution of a monotonic program results in return to the initial coding pattern at completion, nonmonotonic systems can wind up with code that differs from the initial state precisely as a result of execution. The greater the deviation, the lower the degree of resemblance is to the original version that was deposited and, therefore, the less meaningful this deposit. A program that cannot offer a basis for comparison at any point in time stands to lose copyright protection.[14]

From a purely conceptual viewpoint, the danger might appear minimal. The product of self-modification would undoubtedly qualify as a derivative work falling within the original author's ownership. The copyright laws have also been modified so that registration and deposit are no longer indispensable. Although the 1976 Act offers certain procedural advantages to authors who register their works, the power to sue for infringement attaches at the moment of fixation.[15] Hence, all successive phases of a dynamic program's existence are theoretically covered as long as they remain in existence for more than a fleeting instant.

As a practical matter, however, potential difficulties abound and will increase with software power. The simplest nonmonotonic systems, which might alter search or inference patterns in response to a new user-specified fact, present the fewest questions. For example, an expert system based on a relatively limited number of rules could be programmed to save discarded rules and record the source of the modification. This capability would permit tracing back to the original copyrighted version. The prospect of coauthorship by the user (see Copyright Law: Joint Authorship) would also not present an issue. It is unlikely that a program user could appropriate the designer's copyright based on a contribution to the resulting new program form; merely supplying new information can hardly be considered an act of authorship. The interactive, stepwise manner of knowledge enhancement would also permit an owner to re-register the program after each successive modification.

Substantial quandaries surface as user participation increases or knowledge acquisition becomes an automatic feature of program operation. Autonomous data gathering, although clearly a future objective, offers the possibility of software that evolves like an independent organism. Consider a self-updating expert system that monitors developments in relevant technical fields without assistance, determines when incorporation of new information into the knowledge base appears necessary, and performs the required internal accommodation directly. Unless each stage is separately preserved somewhere, how could an infringement action be maintained? More precisely, how could the copyright owner overcome an infringer's defense of independent creation when even a slavishly copied version cannot be readily traced to the original copyrighted program? Copyright protection might well elude fluid programming structures unless coding style is sufficiently distinctive, or complete program records are tenaciously maintained.

Copyright Law: Joint Authorship

Research is progressing in the fields of computer-aided software engineering (CASE), computer-aided design (CAD), and computer-aided manufacture (CAM) at a sufficiently encouraging rate that the prospect of computers producing works of independent, original authorship no longer appears conjectural. Nonetheless, no anticipatory provisions were included in the 1976 Act or 1980 amendments because CONTU considered AI too primitive an area of computer science to warrant concern.[16] CONTU viewed computers as "inert instruments" incapable of producing output more original than the generative instructions entered by a human programmer. This conclusion was recently criticized by another congressional body, the Office of Technology Assessment (OTA). In a 1986 report which exhibits a more sophisticated awareness of technological potential than that of CONTU, OTA stated:

> It is misleading, however, to think of programs as inert tools of creation, in the sense that cameras, typewriters, or any other tools of creation are inert. Moreover, CONTU's comparison of a computer to other instruments of creation begs the question of whether interactive computing employs the computer as co-creator, rather than as an instrument of creation. It is still an open question whether the programmed computer is unlike other tools of creation.[17]

Unless Congress reconsiders the issue soon, it will be up to the courts to decide whether and to what extent the copyright laws apply to computer-generated works.

The output of most current-day computer programs does not raise copyright issues. Today's mass-marketed software simply effectuates user commands in the mechanical fashion expressed by CONTU, operating as sophisticated electronic tools; little originality or creativity is contributed by program operation. However, questions of derivative ownership will arise as software becomes capable of producing integrated, reusable solutions to user-defined problems based on less and less input data. Who should own the copyright to the software output of a CASE system, for example, when the role of the user is reduced to that of mere problem specification, the system itself developing the necessary algorithms and formulating program logic?[18]

The initial task in approaching this question is to locate the threshold of authorship. At what point do computer-generated contributions deserve copyright protection in order to give the program owner rights in software output? Judicial decisions have established two essential requisites of copyright authorship: *originality* and *intellectual labor.* The level of refinement necessary to satisfy both criteria is minimal. For copyright purposes, originality is defined in the narrowest possible sense. Independent creation is all that is necessary; there is no need for the work to be novel. The requirement of intellectual labor is likewise interpreted generously: "[A]lmost any ingenuity in selection, combination or expression, no matter how crude, humble or obvious, will be sufficient."[19]

If courts treat computers like people, alteration and processing of user input will satisfy the copyright notion of authorship when the computer program's contribution is somehow nontrivial—to paraphrase one court, when the variation is something recognizably "its own."[20] Determining whether something sufficiently new has been produced from a precursor presents an innately subjective question. The smaller the resemblance between input and output and the greater the utility of the latter compared to the former, the more likely a court is to recognize a contribution of authorship.

Of course, no guarantee exists that this analysis will be applied to computer output. Courts might simply decide that machines are incapable of creativity or that Congress did not intend the Copyright Act to apply to nonhumans. The latter argument is easily turned around: The statute does not explicitly require human authorship either. Although the text speaks of necessary characteristics for a work to qualify (eligible category of expression, tangible format, and originality), it is silent about the characteristics of those who produce such works.

Whether creativity can or should be attributed to a computer is a much more philosophical issue. If this normative question is to be approached from a perspective short of the metaphysical, analysis should focus on policy: Is computer copyright a socially desirable idea? The protection afforded by copyright law encourages investment of time, labor, and capital in certain forms of expression. Although psychological hesitance to recognize computers as authors might furnish the basis for the denial of copyright to computer-generated works, the effects of such a decision will be felt in terms of incentive. After all, the benefits of copyright flow exclusively to the program's author rather than the obedient machine. Any decision about computer copyright should be viewed as an allocation of financial incentive to those who stand to profit from the law's protection—the program authors and users whose combined efforts produce a finished output.

Let us assume that courts take the plunge and recognize computer authorship. How should ownership rights in computer output be apportioned between the user and the program owner? The short answer is to let them decide for themselves. In a perfectly efficient market, rational economic actors are supposed to arrive at a distribution that reflects the respective values of each party's contribution. The short response to this argument is that some default standard must exist in case the parties forget or for some reason elect not to settle the issue by contract. Furthermore, differences in bargaining power that exist in the real world might obstruct the formation of economically efficient allocations; some approach to measuring "reasonable" apportionment is necessary if only to determine whether a given contract is unconscionable (and hence unenforceable).

Clearly, courts must follow some sort of presumption; but cogent arguments can be made for just about any position and attempts to define a standard will undoubtedly produce heated debate. One view would hold that all claims to computer output rightfully belong to the program's owner, leaving the user with no more than the rights possessed in the parent software. These rights will most probably take the form of a non-exclusive license that permits use but not resale or profit from reuse. A line of support for this sentiment comes from the protection already accorded to derivative works. If output is viewed as a derivative of the creative software, the owner's original copyright will be broad enough to encompass it. The persuasiveness of this reasoning, however, depends on the parent program's mode of operation. By statutory definition, a derivative work must be "based upon" a preexisting work; hence, it must bear a substantial resemblance to its source. Programs that select solutions from a fixed array of preloaded (and presumably copyrighted) templates stand a much better chance of producing derivative output than those which formulate solutions from scratch; the former more obviously give up a "piece" of

themselves to yield a basis for comparison. Another rationale in favor of the parent program's owner is that fragmented ownership rights could result in considerable interference with the owner's ability to license his or her work: A potential infringement lawsuit would arise each time the parent program produced a work substantially similar to one generated for another user.[21]

An opposite approach might insist on according full rights in program output to the user. Licensing of the parent software might be compared to the retention of an employee by the licensee, its price representing a lump-sum payment for unlimited future services. Section 201(b) of the 1976 Act provides, "In the case of a work made for hire, the employer or other person for whom the work was prepared is considered the author for purposes of this title." Alternatively, one might draw an analogy to cases that have arisen in the context of videogames. Despite the fact that players can create a unique series of audiovisual images through their operation of a videogame, courts have refused to consider players the authors of such images because the number and character of all potential displays are ultimately dictated by the game's software.[22]

The position most consistent with the fundamental objective of copyright law—the promotion and protection of authorship—endeavors to apportion ownership according to substantive contribution. Depriving computers of rights in output is equivalent to viewing the program author's contribution as too remote to warrant reward. Approached technically, it seems anomalous to lump all computer software into the inert category of word processors and calculators. Approached functionally, if the success of copyright law can be validly measured by the number of works actually created, it is difficult to see why exclusively favoring either programmer or user would achieve greater overall production of authored output.

Although perhaps noble in purpose, apportioning copyright also demands the most penetrating analysis: How might one begin to identify the substantive elements of authorship, much less quantify the respective donations of two possible authors? Plainly, a rule allocating full ownership either to the program owner or the user would prove far simpler to apply.

Although admitting that difficulty and ambiguity are to some extent unavoidable in determining respective degrees of authorship, a principled basis for analysis does seem to exist. Its crux lies in the idea-expression dichotomy.

Future CAD/CAM or CASE systems will undoubtedly be so advanced that they require from the user only a general sketch of the problem, independently furnishing all levels of the necessary engineering support. Today's highly interactive systems, however, require the user to supply ingredients of the design process to varying degrees. For example, typical CASE systems query the user for a basic algorithmic structure, parameters, and sometimes even labels. The question of authorship contribution requires an understanding of the relevant engineering discipline in order to separate the problem-specification phase, which involves the interplay of ideas, from the solution or design phase that actually results in the production of a tangible, copyrightable work.

Problem specification includes a complete description of the task and extends to the intrinsic structure of the proposed solution. In a world without computers, neither a human engineer nor the engineer's client could claim any rights to such abstract conceptions under copyright law. Copyright protection begins to attach when the hypothesized approach is reduced to a form sufficiently practical that it can be embodied in a tangible work. To return to the CASE example, the art of computer programming can be broken down into several more or less sequential steps: (1) problem identification, (2) algorithmic formulation (if the problem is mathematic in nature) and basic program description, (3) flowcharting and development of a logic structure, (4) arrangement of subroutines and program modules, (5) structuring of data components, and (6) coding. Of these steps, the first two most clearly involve ideas; the remaining are closely tied to the source code product, expressing individual design choices of the programmer, and should be protectible.[23]

Once protectible elements of the engineering process have been identified, a weighted value must be assigned to each stage and the respective contributions of user and program quantified. Both come down to a difficult matter of scientific and economic judgment. The proportionate value of an accomplishment within a particular defined category in relation to the overall project would best be measured by the current degree of technical difficulty inherent in this category. That is, a court might utilize expert testimony to help appraise the relative amount and worth of equivalent human labor currently necessary to accomplish each subtask protectible under copyright law. A court might assess the source of accomplishments within a category through inspection of the parent program's input and output, evaluation of its capabilities, and comparison of the final product to any templates embedded in the parent's structure.

Not an easy task, to be sure, but courts assess damages in complex and factually ambiguous circumstances all the time. As the Supreme Court once put it, "[C]ases will often occur in which it is evident that large damages have resulted, but where no reliable data or element of certainty can be found by which to measure the amount.... To deny the injured party the right to recover any actual damages in such cases, because they are of a nature which cannot be thus certainly measured, would enable parties to profit by, and speculate upon, their own wrongs.... Such is not, and cannot be, the law[.]"[24]

Patent

Because the patent system was created to offer inventors protection for their useful ideas, one might naturally suppose that it picks up where copyright protection leaves off. Such is by no means the case, however, for two different reasons.

First, the character of protection

The Supreme Court ... recognized that permitting a single individual to obtain control over an entire phenomenon of nature would constrict, rather than promote, the progress of science....

afforded by a patent and the means of obtaining it differ drastically from the simple registration procedure and the longevity of copyright. Acquiring a patent takes a great deal of time: Two to five years can elapse between the filing of an application with the Patent and Trademark Office (PTO) and the date the patent is finally issued. Software can become obsolete during this long wait, and the Patent Act (Title 35 of the U.S. Code) provides the applicant with no right to exclude practice of an invention by others in the interim. The lifespan of patent protection is also relatively short: The maximum period is 17 years. In contrast, for works created on or after 1 January 1978, a copyright extends over the entire life of the author plus 50 years. Finally, patents are expensive: PTO application and maintenance fees over the life of a patent amount to over $3000, and the legal services ordinarily required for successful prosecution can cost tens of thousands of dollars. However, the scope of protection once achieved is formidable: A patent owner possesses the right to exclude others from using, making, or selling the invention, whereas copyright protection extends only to the exclusive right to copy and trade secret protection to an exclusive right to use.

Second, the Patent Act does not protect ideas by themselves. One reason for this important limitation is embodied in the requirement of utility. The law will grant an inventor a virtual monopoly over his or her creation only if the public stands to receive some corresponding benefit from the invention's usefulness. An idea for a potentially useful article or process has no utility of its own; the Act does not recognize value until this precursor is reduced to a form of actual usability. Hence, disembodied figments of cognition are no safer under patent law than copyright law.

In addition to utility, the Patent Act specifies three further criteria of acceptability for protection under its terms: The invention must (1) fall within the subject matter of the Act's authority, and meet the interrelated tests of (2) novelty and (3) nonobviousness. Most unsuccessful software applications have been rejected for failure to state statutory subject matter.

Patent Law: The Subject Matter Test

Section 101 defines permissible subject matter to include any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof. In a series of three opinions delivered over the course of a decade, the Supreme Court has defined the parameters of patent protection for computer software based on its interpretation of this section. In Gottschalk v. Benson,[25] the Court first announced the principle that a computer program whose function is to solve a mathematical algorithm fails under section 101. Subsequent opinions in Parker v. Flook[26] and Diamond v. Diehr[27] qualified the ban on algorithmic programs to permit patenting of larger processes that utilize a mathematical algorithm as part of their operation. In Diehr, the invention was an integrated system for curing synthetic rubber in a mold with heat. Proper cure times were calculated according to an equation discovered by the Swedish scientist Svante Arrhenius. Five out of nine justices viewed the overall process as patentable despite the presence of mathematics because once arrived at, cure times were translated directly into the mechanical control of the press. Viewing the overall invention, the majority explained, "[A] claim drawn to subject matter otherwise statutory does not become nonstatutory simply because it uses a mathematical formula."

This reasoning does not mean, however, that recharacterizing the software process as a hardware apparatus merely because a computer is involved will serve to circumvent the Court's rule. The larger apparatus must perform some independent function beyond the generation of solutions.[28] Lower courts—primarily the Court of Customs and Patent Appeals (CCPA), recently renamed the Court of Appeals for the Federal Circuit—have consistently applied the same criteria to all inventions regardless of the form stated on the application.

In one well-known case, for example, the inventor of a computerized sales organizational model attempted to patent the entire system as a uni-

fied package, hoping to avoid examination of the mathematically based software. The Court saw through this strategy and rejected the application under section 101, stating, "Labels are not determinative in section 101 inquiries ... 'because the form of the claim is often an exercise in drafting.'"[29]

The Supreme Court's refusal to permit patenting of mathematical algorithms is based on a deeper principle of patent law that apart from the criterion of utility, presents a further barrier to the patenting of ideas. In a famous controversy that occurred over a century ago, Samuel Morse applied for a patent on all types of communication utilizing electromagnetic transmission. The Supreme Court rejected his application.[30] It recognized that permitting a single individual to obtain control over an entire phenomenon of nature would constrict, rather than promote, the progress of science. Even if it possesses inherent utility, an idea can simply cover too broad a swath of potential technology. The Court considered this danger sufficiently great to justify a strict prohibition: Ideas themselves are simply never patentable subject matter.

The effort to define a critical level of scientific basicness has confounded courts ever since. The Benson majority seized on the mathematical algorithm as a means of identifying this threshold for computer software, in effect holding mathematical expressions and algorithmic representations to be equivalent to scientific principles and ideas. The Court further included "mental processes" and "abstract intellectual concepts" in its lexicon of disfavored notions, stating that "they are the basic tools of scientific and technical work."

The Supreme Court's rules can be summarized as follows: (1) If a computer program does not contain a mathematical algorithm or embody a mental process or abstract intellectual concept, there is no subject matter bar under section 101. (2) If one of these elements is discovered, the software is presumptively unpatentable; however, its incorporation into an otherwise independently patentable process or apparatus does not destroy the latter's patentability. (3) In making the second determination, a court must look to the substance, rather than the form, of the application.

Few cases reach the Supreme Court. Most patent controversies are instead resolved by internal PTO review and appeal processes or in a lower federal court. Although the Supreme Court might write the rules, the manner in which PTO and the courts share oversight of the patent system can produce consequences of its own.

When an inventor submits a patent application, it is reviewed by PTO examiners according to the current law, as set forth in the patent statutes and cases interpreting these statutes. Should PTO issue a final decision denying a patent, an applicant willing to bear the time and expense can seek further review in the Federal Circuit. The numerous decisions of this federal tribunal and its predecessor enrich the legal framework in which applications are judged and give practical meaning to the Supreme Court's broad edicts. If the court's view of the law is more expansive than that of PTO, it will reverse PTO and grant the inventor a patent; if not, it will affirm. It is only after all these review levels have been exhausted that the applicant can petition for a hearing before the Supreme Court.

Denial of the application is not the only way in which validity of a patent can be tested in federal court. A party who has been sued for infringement might raise invalidity of the patent as a defense. Alternatively, someone who is nervous about a possible infringement lawsuit can commence a declaratory judgment action to test the patent's effectiveness before trouble begins. In these circumstances, a court's discretion is limited; judges are not as free to erase patents already granted by PTO as they are to approve applications that PTO has denied. The Patent Act presumes a patent to be valid and casts the burden of showing otherwise on the party so asserting.[31] Nonetheless, applicants who have achieved success with PTO must live with the knowledge that their rights are not immune from challenge. Judges will ultimately have the last word if called on to act. Differing views held by PTO and the courts can result in an uncertain future for patent holders, a future that might well be determined by the fortuity of a lawsuit.[32]

To appreciate the ways in which this legal and regulatory environment can affect the patentability of AI systems, consider the recent series of applications prosecuted by Teknowledge Inc. Teknowledge successfully obtained patents for its computer-integrated manufacturing (CIM) expert system and two expert system-development packages. PTO considered all three programs to be patentable subject matter. Would a court agree? Maybe, maybe not. Will the issue ever come before a court? Who knows.

If a court were to someday evaluate the validity of any of the Teknowledge patents, it would first have to determine whether the program satisfies the section 101 criteria. As an example, the CIM system assists in the manufacture or assembly of complex modular products by generating a design configuration that is consistent with customer needs as well as marketing and engineering constraints. The system is flexible, permitting alteration of the constraints to accommodate product updates or modifications. Does this software state a mathematical algorithm? The answer depends on one's definition. If an algorithm is viewed as a set of procedures performed on a given input to produce a desired output, then no computer program is safe: All operations performed by digital technology ultimately reduce to manipulations of numeric values, and all goal-directed tasks can be characterized as algorithmic. This interpretation, which had once been urged by PTO, was explicitly rejected by the Supreme Court in Diehr. Instead, the Court described the forbidden fruit as "a procedure for solving a given type of mathematical problem ... or [a] mathematical formula." By drawing this relatively narrow definition, the Court undoubtedly believed itself making life easier for patent-seeking programmers while preventing monopolies on "laws of nature." Regrettably, its choice of words seems too ambiguous to have accomplished either aim. Numbers are simply too ubiquitous in comput-

er science. With the exception of software whose function is limited to rearrangement of character strings, virtually all computer applications involve some degree of quantitative analysis. Does mathematics mean anything involving information that is represented numerically, or does Diehr forbid patenting only of pure methods of calculation or fundamental scientific relationships expressed as mathematic equations?

The latter might be the direction taken by the Federal Circuit[33] and would probably protect CIM from disqualification as a mathematical algorithm. One perhaps might view the transformation of customer input data into a manufacturing blueprint through the application of definitions and constraints as mathematical, that is, if logic operations are to be included under this heading along with numeric calculations. But Teknowledge patented more than a sorting device. The key feature of the CIM system is its transparent representation of control knowledge. Users can vary more than just input values; the constraints themselves can be altered to accommodate entirely different product hierarchies or even knowledge domains. The essence of Teknowledge's invention, therefore, is not the solution to a particular problem but a means of formulating knowledge-based solutions to an entire class of problems. If anything, CIM is a creator of algorithms; mathematic processing is a result rather than the objective.

The remaining section 101 hurdle, far more threatening to the AI community in general and to CIM in particular, is the Court's antipathy toward "mental processes." The logical focus of a test based on cognition resides in the capability of the human brain. Quite ironically, then, the more sophisticated the system—that is, the more it captures attributes previously thought accessible only to human cognitive processes—the more the mental process exclusion appears satisfied.

Two relatively recent cases illustrate the danger. CCPA sanctioned as statutory subject matter a software system designed to perform natural language translation in In re Toma.[34] The court took pains to assert that no thought processes were involved in the particular method of translation, which involved simple data manipulation, thereby implying that the **form** of reasoning employed by the program might hold sway. Even this unfortunately restrictive view seems to have been discarded in In re Meyer,[35] which concerned a forerunner of current-day expert systems. In this case, the applicant's program was designed to accumulate and analyze the results of a series of diagnostic test results in order to narrow the possible sources of a problem. Although the system appears to have been designed for general application, the operational example selected by the applicant was that of a neurologist employing the system to coordinate the results of various physical tests in aid of medical diagnosis. The court rejected the application under section 101. It made no attempt to discern whether the applicant's process followed reasoning patterns similar to those which might be employed by a human neurologist, holding simply that transfer of a cognitive **task** to computer control is sufficient to render the software unpatentable.

Does CIM implicate a mental process? Teknowledge was careful in its patent application to avoid any intimation that CIM mimics a human intellect. The description of the invention is purely functional:

> ...a hierarchical knowledge base comprising a decomposition of a set of elements into subsets over a plurality of hierarchical levels, a plurality of respective predefined functions or conditions of the elements within the subsets at a plurality of the hierarchical levels, and a predefined set of operations to perform on a user-defined set of elements responsive to the knowledge base.[36]

If courts treat the mental process exclusion as a literal mandate to reject patent applications whenever such processes are detected, AI practitioners will face increasingly hostile treatment as their work grows more successful. Such a result directly contradicts the policy of a system that exists to promote scientific advance. What arguments might be made against such an ironic result? The most direct rebuttal would confront the wisdom of adopting the mathematical algorithm as a surrogate for ideas. Although abandoning this rule might seem desirable as a pure matter of policy, such an approach demands an uphill fight against the practical obstacle of established precedent. A second, less ambitious crusade would attempt to limit the range of propositions included under the disfavored term. Effective advocacy secured rejection of the PTO's original definition of mathematical algorithms, which was expansive enough to encompass every data processing operation simply because digital computers function by manipulating binary-coded numerals,[37] in favor of the narrower view stated earlier. The concept of mental processes might await similar judicial narrowing.

With hope, courts will be persuaded that a patent covering the computational repetition of a thought process does not imply exclusive control over the thought process itself. A cognitive task is not the same as a scientific principle. The law of gravity, to take one example, is self-operative; it cannot be improved or replicated on a machine. The mere fact that the manifold abilities of the human brain were harnessed long before computer implementation became feasible should not operate as a bar to protecting the most valuable computer software systems.

Patent Law: Novelty and Nonobviousness

Section 101 defines the universe of patentable subject matter. Determining whether a patent ought to cover a particular invention is the province of two further statutory tests, those of novelty and nonobviousness. The novelty requirement is stated categorically in section 102, which specifies only that the invention must somehow differ from methods and objects already in the prior art. Section 103, cast in the form of a negative proscription rather than an affirmative requirement, articulates the increment of advance over existing subject matter necessary to justify patent protection:

> A patent may not be obtained … if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains.

In Dann v. Johnson,[38] the Supreme Court was presented with a data processing system capable of providing bank depositors with a detailed periodic statement, segregating transactions into different categories selected by the customer. Noting that a depositor could achieve the same result by maintaining a series of separate accounts, the Court concluded that the combination of widespread use of computers in the banking industry and the existence of a previously issued patent involving similar functions rendered the claim obvious. The message of Johnson is that programs which simply computerize a known process on existing machines through familiar techniques are too unremarkable to deserve patent protection.

Few existing computer programs could claim credit to a previously unknown process because the vast bulk of programming efforts are directed at automating mundane tasks. It would be difficult to argue, for example, that current bookkeeping software has actually enhanced the current state of the art in the field of accounting. AI programs that capture high-level cognitive procedures might present a more compelling case. If the reasoning process is not easily explained or reproduced without the aid of AI techniques, the process, although familiar, might not be considered "known."

Distinguishing new machines from old ones used in new ways presents the classic patent law question of what constitutes inventive difference. The analysis ultimately reduces to an assessment of utility—whether the new difference makes a difference—but the difficulty of drawing such subtle distinctions has led to a highly technical set of substantive criteria. Their enumeration is beyond the scope of this article, and it seems safe to conclude only that the law is unsettled and guiding principles few.

Some intellectual activities that form the basis of AI research might, through this research, become well enough understood to be "known." At the same time, few pure software systems require special hardware modifications for their execution. Until AI programs based on known mental processes become so superior to their human counterparts as to be capable of independently adding to the state of knowledge (and thus furthering the art) in a given domain, the most likely route to patentability lies in convincing PTO and the courts that the software represents an advance to the art of computer programming. The touchstone of nonobviousness is innovation. Integrated systems that combine special-purpose hardware devices with control software, such as computer-based vision or robotics applications, stand the best chance of patentability as independent apparatus. A "new" machine is most clearly present (and the rationale of Johnson least applicable) when hardware and software are expressly designed for the cooperative execution of a single task.

Trade Secrets

The patent and copyright laws promulgate disclosure systems. Protection is bestowed as a means of encouraging the free exchange of ideas and information. Trade secret law performs quite the opposite function: It assists active efforts to maintain confidentiality. The primary responsibility for creating and enforcing a scheme to restrict the flow of proprietary information lies with the party seeking protection; the law operates only as a second line of defense.

The purpose of trade secret law is to maintain proper standards of commercial ethics rather than to protect the objects of secrecy. Parties entering into confidential business relationships must remain faithful to their promises of nondisclosure. Outsiders are prohibited from purloining the efforts of businesses to attain competitive advantage, particularly when valuable resources have been expended in doing so.

The law of trade secrets protects against improper disclosure or wrongful appropriation by affording an aggrieved party the opportunity to sue for a prohibitory injunction (which prevents the defendant from making use of the secret) or monetary damages. Injunctions are commonly issued for at least the length of time that would be required to develop the device or process without access to the trade secret (that is, the reverse engineering period), and possibly longer if the defendant's conduct is considered particularly scurrilous.[39]

The scope of trade secret law is shaped both by its functional orientation and its source of enactment. Trade secret protection is a creature of state law. Unlike the federal Copyright and Patent Acts, whose terms are national, the law of trade secrecy can vary from state to state. One widely followed statement defines its coverage as applying to information, including a formula, pattern, compilation, program, device, method, technique, or process, that derives independent economic value, actual or potential, from not being generally known to, and not being readily ascertainable by proper means by, other persons who can obtain economic value from its disclosure or use.[40] The focus of trade secret law on commercial conduct, rather than intrinsic properties of the secret itself, results in a flexible standard of acceptable subject matter. Computer software has been held protectible when sufficient originality exists to produce a competitive advantage.[41]

The boundaries of protection in a given instance are set by the interplay between precautions taken to prevent disclosure and the manner in which information is obtained without authorization. Simply put, trade secret law protects nonpublic knowledge. If the secret is revealed indiscriminately, either deliberately or through failure to take reasonable preventive steps, protection disappears. Minor leakage, however, only weakens protection: If the trade secret is procured by wrongful means, the defendant might still be held liable. No amount of precaution, however, can protect a secret from legitimate reverse engineering.

Businesses have made use of a number of devices and procedures to pro-

If courts treat the mental process exclusion as a literal mandate to reject patent applications whenever such processes are detected, AI practitioners will face increasingly hostile treatment as their work grows more successful.

tect trade secrets. The most obvious concern of software developers is misappropriation by a program licensee, but trade secrets can also be lost to employees, partners, joint venturers, and competitors who resort to industrial espionage. The following measures have proven effective in protecting trade secrets.

Restrictive licenses with users: The most reliable licenses are those which can be negotiated directly with the licensee. Licenses attached to mass-produced software that reaches the user through the marketplace are not only difficult to enforce as a practical matter but might also lack legal standing.[42]

Notice provisions: Broad dissemination of material that embodies trade secrets is not necessarily fatal to protection. If recipients are put on notice that the resources are being shared on a confidential basis, unauthorized use or disclosure can be stopped when discovered. Clear proprietary legends and confidentiality restrictions affixed to documents or made to appear on program output generally provide the requisite notice.

Nondisclosure agreements: Many courts consider a duty not to breach the employer's confidences to be an implied condition of any employment relationship. Nevertheless, an express written agreement often proves useful. Not only will explicit language eliminate any ambiguity about what information is to be kept confidential, but it can also create rights against the pilferage of knowledge that falls short of being a trade secret.

Although software has been held protectible as a trade secret, certain conceptual difficulties can impede the law's applicability to AI systems. If a computer program's function is to reproduce processes that routinely go on in the minds of human beings, perhaps there is no secret to protect. This view might appear particularly inviting in the case of expert systems. Courts might be tempted to distinguish between shallow-level and deep-level systems as a means of identifying programs that contain novel or unfamiliar elements. A shallow expert system is one that contains only empirical knowledge, associating various input states directly with actions. Shallow systems draw their capability directly from the experience of human experts as revealed by knowledge engineering. In contrast, deep systems represent concepts at a basic causal level and base conclusions on some theoretical model of the underlying discipline.[43] Such special programming features as a self-updating ability or the capacity for adaptation might also create the impression that a system rises above the sources tapped for expertise.

Militating against this approach based on appearances is the fact that a working knowledge base requires labor and skill to produce and secures an undoubted competitive advantage. Specialized programming architectures might also prove necessary for efficient operation. Courts that have approved of trade secret protection for computer software have focused less on program attributes or innovation than commercial usefulness and feasibility.[44] For many applications (particularly when operational flexibility is not critical), shallow systems, whose reduced complexity typically results in lower startup costs, are considered more suitable than deep systems. On an abstract plane, the patent law notion that mental processes can be familiar without being intrinsically understood also operates in favor of protection.

From a practical standpoint, protection through trade secrecy is well suited to AI system development. AI software is typically produced for a single user or a narrow class of users, permitting the issuance of licenses on an individualized basis. Low sales volume also permits realistic enforcement: Violations are relatively easy to discover and can be traced to a particular user. The high degree of program sophistication further enhances protection by making permissible reverse engineering (or illegitimate decompilation) more difficult to accomplish. Perhaps most importantly, the haphazard and largely uncertain scope of federal protection under the copyright and patent laws can leave AI researchers without any realistic alternative.

In an ideal world, these two federal systems would serve as complementary protection alternatives—copyright for unique forms of program expression and patent for inventive problem-solving methodologies. Currently, the jury remains out as to copyright, and courts have not displayed a great deal of enthusiasm toward software patents.

This state of affairs is most unfortunate. Persistent reliance on the law of trade secrets might well prove detrimental to the AI industry over the long run. In many emerging areas of

AI technology, development proceeds at the level of basic research. The sharing of ideas is necessary if incremental progress is to be made without needless duplication of effort. This problem is far smaller for developers of today's commercially distributed software. Because programming is simpler and application techniques widely known, maintaining secrecy of source code might not impede technological advance. Function seems sufficiently related to form that even superficial knowledge of a program's capabilities will avoid much duplicative effort. In contrast, entirely new paradigms and system architectures might be necessary to achieve even modest AI goals. Little chance seems to exist of deducing models of cognitive processing from program descriptions or even hands-on operation. Comparable efforts will likely prove too few and the general state of knowledge too undeveloped for meaningful insights to be drawn, cooling the pace of research unnecessarily.

Notes

The footnote style employed here is a specialized citation form peculiar to legal materials. This style has been developed to maintain consistency among jurisdictions, case reporters, and statutory citations. Its basic format is: <volume> <SOURCE> <first page of document>, <local citation>.

1. The copyright clause of the U.S. Constitution empowers Congress to enact laws protecting the "writings" of an "author." U.S. Const., art. I, section 8, cl. 8. A number of federal copyright acts have been legislated under this authority.

2. 17 U.S.C. section 102(a).

3. Indeed, courts had for some time recognized the useful article doctrine, which excludes from copyright protection the design or shape of an article unless these qualities can be separated from, or exist independently of, the article's utilitarian aspects. The purpose of this exclusion is to prevent inventors from circumventing the rigorous standards of the patent laws by simply copyrighting their designs.

4. Although the mechanics of obtaining copyright and patent protection are beyond the scope of this article, a good source of general information is Bender, D. 1987. *Computer Law*. New York: Matthew Bender.

5. 17 U.S.C. section 102(b). This statement is really no more than a generalization of the useful article doctrine. Ideas are free to all, and the public is entitled to profit from the ideas contained in copyrighted works. The author of a cookbook, for example, cannot prevent readers from preparing the recited recipes.

6. Morrissey v. Procter & Gamble Co., 379 F.2d 675 (1st Cir. 1967) (sweepstakes rules). Some courts will permit infringement actions in such instances but only when the copying is slavish. *See, e.g.*, Continental Casualty Co. v. Beardsley, 253 F.2d 702 (2d Cir. 1958).

7. 714 F.2d 1240 (3d Cir. 1983), *cert. denied,* 464 U.S. 1033 (1984).

8. A key aspect of this decision was the court's ruling that the existence of a limited number of ways to structure or arrange such an operating system will not merge expression with the idea; even similarly structured programs can be expressed differently through alternative coding choices.

9. Whelan Associates, Inc. v. Jaslow Dental Laboratory, Inc., 797 F.2d 1222 (3d Cir. 1986), *cert. denied,* 107 S.Ct. 877 (1987).

10. 659 F.Supp. 449 (N.D. Ga. 1987).

11. 17 U.S.C. section 102(a).

12. 17 U.S.C. section 101.

13. McDermott & Doyle, *Non-Monotonic Logic I,* 13 Artificial Intelligence 41, 44 (1980).

14. This problem in its most rudimentary form is currently under debate within the Copyright Office. Proprietors of commercial databases, which constantly expand their systems to include the latest information, have sought a means of complying with copyright registration requirements without the need for continued redeposit. The Copyright Office has recently issued proposed regulations for group registration of a single database and updates occurring over a three-month period. *See* 52 Federal Register 37167.

15. Registration is a technical prerequisite to actual filing of an infringement action. 17 U.S.C. section 411. However, there is no absolute requirement that the work be registered at the time infringement takes place. Registration made just prior to the lawsuit will open the courthouse doors, even though the alleged infringement occurred long before.

16. CONTU noted "concern that computers either had or were likely to soon achieve powers that would enable them independently to create works that, although similar to other copyrightable works, would not or should not be copyrightable because they had no human author. The development of this capacity for 'artificial intelligence' has not yet come to pass, and, indeed, it has been suggested that ... such development is too speculative to consider at this time." Final Report of the National Commission on New Technological Uses of Copyrighted Works, July 31, 1978, at 42.

17. U.S. Congress, Office of Technology Assessment, Intellectual Property Rights in an Age of Electronics and Information 72 (1986).

18. Note that a similar question might someday be raised in the patent arena by sophisticated computer-aided design programs that contribute to the design of inventions. However, because the standard of inventiveness set forth in the patent laws is far more demanding than the originality necessary for copyright protection, the likelihood of a court recognizing the contribution of a computer program is proportionately more remote.

19. Nimmer, M. 1987. *The Law of Copyright,* section 1.08[C][1]. New York: Matthew Bender.

20. L. Batlin & Sons, Inc. v. Snyder, 536 F.2d 486, 487 (2d Cir. 1976), *quoting* Alfred Bell & Co. v. Catalda Fine Arts, Inc., 191 F.2d 99, 103 (2d Cir. 1951). The case concerned a human author, and the court's actual words were "his own."

21. An analogous issue arises when an author assigns to a publisher the copyright to the author's manuscript. Should the author then produce a substantially similar work for another audience, the publisher can sue for copyright infringement.

22. *See, e.g.,* Midway Manufacturing Co. v. Artic International, 704 F.2d 1009 (7th Cir. 1983), *cert. denied,* 464 U.S. 823.

23. The House Report accompanying the 1976 Act stated that "the expression adopted by the programmer is the copyrightable element in a computer program.... [T]he actual processes or methods embodied in the program are not within the scope of copyright law." H.R. Rep. No. 1476, 94th Cong., 2d Sess. 51, 57, *reprinted in* 1976 U.S. Code Cong. & Ad. News 5659, 5670. The Whelan and DCA cases, however, indicate that copying the organization and structure of a program might amount to infringement even if no large segments of code are actually reproduced.

24. Story Parchment Co. v. Paterson Co., 282 U.S. 555, 564 (1931).

25. 409 U.S. 63 (1972).

26. 437 U.S. 584 (1978).

27. 450 U.S. 175 (1981).

28. For example, in In re Bradley, 600 F.2d 807 (CCPA 1979), the court reversed the Patent and Trademark Office's rejection under section 101 of a firmware system designed to facilitate internal data transfers between the central processing unit

ANNOUNCING...

ARTIFICIAL INTELLIGENCE FRONTIERS

Edited by B. Chandrasekaran, *Department of Computer and Information Sciences, Ohio State University*; C. Rich, *Artificial Intelligence Laboratory, Massachusetts Institute of Technology*; A. Meyrowitz, *Office of Naval Research*

A new series of monographs, multi-authored works, and edited treatises representing the forefront of research in machine learning, natural language processing, distributed problem solving, intelligent and expert systems, knowledge representation, human-computer interaction, intelligent tutoring systems, logic programming, and other areas of artificial intelligence.

CLASSIC TEXTS —

Introduction to Logic Programming
Christopher J. Hogger
1984, 288 pp. $60.00 Casebound/ISBN 0-12-352090-8
$29.50 Paperback/
ISBN 0-12-352092-4

Symbolic Logic and Mechanical Theorem Proving
Chin-Liang Chang and Richard Char-Tung Lee
1973, reprinted 1987, 337 pp.
$29.95 Casebound/
ISBN 0-12-170350-9

Introduction to Common Lisp
Taiichi Yuasa and Masami Hagiya, Translated by Richard Weyhrauch and Yasuko Kitajima
1987, 280 pp. $29.95 Casebound/ISBN 0-12-774860-1

Common Lisp Drill
Taiichi Yuasa, Translated by Richard Weyhrauch and Yasuko Kitajima
April 1988, 250 pp. (tentative)
$24.95 (tentative) Casebound/
ISBN 0-12-774861-X

Academic Press
Harcourt Brace Jovanovich, Publishers
San Diego, CA 92101-4311

Credit card orders call toll free 1-800-321-5068
From Missouri, Hawaii, or Alaska 1-314-528-8110
Prices are in U S. Dollars and are subject to change.

San Diego New York Boston London Sydney Tokyo Toronto

41048

For free information, circle no. 30

and the main memory. It reasoned that the applicant did not claim information embodied in the firmware or the firmware itself but rather a combination of dedicated hardware elements assembled to perform a specific task.

29. In re Maucorps, 609 F.2d 481, 485 (CCPA 1979), *quoting* In re Johnson, 589 F.2d 1070, 1077 (CCPA 1978).

30. O'Reilly v. Morse, 56 U.S. 62 (1863).

31. 35 U.S.C. section 282.

32. To illustrate the extent to which the views of the Patent and Trademark Office (PTO) can diverge from those of the courts, the PTO recently granted a patent on what is essentially a computer-implemented mathematical algorithm—precisely what the Supreme Court has unambiguously forbidden. Patent No. 4,646,256, granted on 24 February 1987, sets forth a "special purpose computer" and computational method for performing N-Length, real-number discrete transforms. No postsolution activity is undertaken by the invention on its purely numeric output.

33. *See, e.g.,* In re Pardo, 684 F.2d 912, 915 (CCPA 1982) ("It would be unnecessarily detrimental to our patent system to deny inventors patent protection on the sole ground that their contribution could be broadly termed an 'algorithm'"); In re Meyer, 688 F.2d 789, 795 (CCPA 1982) ("The presence of a mathematical algorithm or formula in a claim is merely an indication that a scientific principle, law of nature, idea or mental process may be the subject matter claimed and, thus, justify a rejection of that claim under 35 U.S.C. section 101; but the presence of a mathematical algorithm or formula is only a signpost for further analysis").

34. 575 F.2d 872 (CCPA 1978).

35. 688 F.2d 789 (CCPA 1982).

36. Patent No. 4,591,983 at 3.

37. Diamond v. Diehr, 450 U.S. 175, 186 (1981).

38. 425 U.S. 219 (1975).

39. *See, e.g.,* Analogic Corp. v. Data Translation, Inc., 371 Mass. 643, 649 (1976), in which the Massachusetts Supreme Judicial Court stated, "The plaintiff is entitled to have its trade secrets protected at least until others in the trade are likely, through legitimate business procedures, to have become aware of these secrets. And even then the defendants should not be permitted a competitive advantage from their avoidance of the normal costs of invention and duplication. Where the defendants have saved substantial expense by improperly using confidential information in creating their product, the ultimate cessation of an injunctive order might well be conditioned on their payment of an appropriate sum to the plaintiff."

40. Uniform Trade Secrets Act, section 1(4).

41. *See, e.g.,* Telex v. IBM, 367 F.Supp. 258, 323 (N.D. Okla. 1973), *affirmed on trade secret issue,* 510 F.2d 894 (10th Cir. 1975), *cert. dismissed,* 423 U.S. 802 (1975).

42. Illinois and Louisiana recently enacted statutes validating "shrink-wrap" licenses. The Louisiana version, however, was promptly struck down by a federal court, which viewed it as an impermissible intrusion into the territory covered by the Copyright Act. Vault Corp. v. Quaid Software Limited, 655 F.Supp. 750 (E.D. La. 1987). The decision has been appealed, but the view of most commentators appears similarly negative. *See* Harris, *A Market-Oriented Approach to the Use of Trade Secrets or Copyright Protection (or Both?) for Software,* 25 Jurimetrics J. 147, 154-56 (1985).

43. The distinction is explained and examples are given in Hart *Directions for AI in the Eighties,* Sigart Newsletter #79 (1982), at 11.

44. *See, e.g.,* Structural Dynamic Research Corp. v. Engineering Mechanics Research Corp., 401 F. Supp. 1102, 1117 (E.D. Mich. 1975) ("An overwhelming majority of authorities in the field do not recognize novelty and uniqueness as eligibility requirements for trade secret protection").