UIC John Marshall Journal of Information Technology & Privacy Law

Volume 9
Issue 3 *Computer/Law Journal - Summer 1989*

Article 3

Summer 1989

Copyright, Patent, and the Virtual Machine, 9 Computer L.J. 321 (1989)

Richard L. Torczon

COPYRIGHT, PATENT, AND THE VIRTUAL MACHINE

. . . quia frustra fit plura quod potest fieri per pauciora. . .†

By RICHARD L. TORCZON*

TABLE OF CONTENTS

† ". . . for that is needlessly accomplished through more which can be accomplished through fewer . . ." WILLIAM OF OCKHAM, DE SACRAMENTO ALTARIS, TERTIO QUAERITUR 104-05 (T. Birch trans. 1930). Occam's Razor (also called Ockham's Law of Parsimony) is usually stated *Entia non sunt multiplicanda praeter necessitatem* which loosely translates, "Do not multiply things unnecessarily." W.M. Thorburn argues that Brother William never actually wrote the later formulation; it was instead coined by the Scotist commentator John Ponce of Cork in 1639. *See* Thorburn, *The Myth of Occam's Razor*, 27 MIND 345 (1918). Thorburn cites Occam's Razor and another common misquote allegedly from Kant to illustrate the danger of relying on university professors for accurate quotes.

* Mr. Torczon is a third year student at the University of Texas, School of Law. He received a B.A. in Computer Science from Rice University in 1986, and recently completed an internship with Adachi International, the largest patent law firm in Nagoya, Japan.

I. INTRODUCTION

Intellectual property is a paradox: it seeks to protect original creations and expressions,[1] but not ideas, concepts, or natural laws.[2] This inherent paradox is further complicated whenever new technology or methods are introduced. Nowhere is this more apparent than in computer intellectual property law.[3] Industry lawyers must rely on con-

1. "The Congress shall have Power . . . To promote the Progress of Science and useful Arts, by securing for limited Times to Authors and Inventors the exclusive Rights to their respective Writings and Discoveries . . . ." U.S. CONST. art. I, § 8, cl. 8.

2. *E.g.*, Funk Bros. Seed Co. v. Kalo Inoculant Co., 333 U.S. 127, 132 (1948) (The court would not allow "a patent to issue on one of the ancient secrets of nature now disclosed."); Gottschalk v. Benson, 409 U.S. 63, 68 (1972) ("Phenomena of nature, though just discovered, mental processes, and abstract intellectual concepts are not patentable, as they are the basic tools of scientific and technological work."); 17 U.S.C.A. § 102(b) (West Supp. 1987) (Copyright Act of 1976).

3. " 'Technology has outstripped our patent and trademark legal system,' says Fred M. Gibbons, president and chief executive of Software Publishing Corp. 'We don't know how to deal with these issues. It's not sheet music, it's not a recording.' " Wall St. J., Mar. 21, 1988, at 17, col. 4.

flicting decisions or on cases involving totally unrelated subject matter for guidance.[4] Worse yet, while the various types of intellectual property protections are quite distinct, the dividing lines between types of computer innovations are often hazy at best.

Computer innovations have traditionally been divided into hardware and software. Hardware is tangible and manufactured, hence it is protected by patent. Software is ephemeral, except when it is printed on a piece of paper or on a video display screen, and is written, so it is protected by copyright. Unfortunately, these divisions are too rigid, too black-and-white. Intermediate objects, called firmware,[5] exist that are neither fish nor fowl. A common example of firmware is the integrated circuits in calculators that allow them to continue to perform complicated functions even after power has been disrupted. Firmware makes more powerful and flexible hardware designs possible by allowing hardware designers to replace some of the hardware with a program. Many components that are widely considered to be hardware are in fact firmware. For instance, most microprocessors, such as Motorola's MC68000, contain both microcircuitry and microprogramming.[6] This confusion over hardware, firmware, and software is natural since they are really part of a continuum. "Hardware and software are logically equivalent."[7] In theory, there is no software, nor could there be, that can be run on a computer but can not be built entirely from hardware.[8] Since computer innovations lie along a continuum, it seems odd that the intellectual property protection they receive varies sharply.

The varied legal treatment of computer innovations has caused much confusion.[9] This confusion could be reduced if the legal model for

---

4. "[L]awyers are left with a muddle of conflicting case law —much of it not concerning software at all. In fact, some lawyers say, the Apple suit itself could well turn on a 1986 decision involving stuffed dinosaurs." *Id.*

5. Firmware is "software embedded in electronic devices (*i.e.*, hardware)." A. TANENBAUM, STRUCTURED COMPUTER ORGANIZATION 11 (2d ed. 1984).

6. *See* MOTOROLA INC., MC68000 16-BIT MICROPROCESSOR USER'S MANUAL (3d ed. 1982).

7. A. TANENBAUM, *supra* note 5. Tanenbaum calls this the "central theme" of his book. *Id.* It is also the central theme of this Article. Logically equivalent means the operations performed by the computer are the same whether they are performed by hardware or by software.

8. Since all work in a computer is a physical response to a series of electrical pulses that the hardware interprets as off or on signals, and since those pulses can be generated by a program or by hardware physically wired to voltage sources representing the same electrical pulses, hardware and software are functionally interchangeable limited only by cost, complexity, inflexibility, and, perhaps for exceedingly complex programs, physical limitations of the current technology. *See* H. HANNEMAN, THE PATENTABILITY OF COMPUTER SOFTWARE 2 (1985); T. PRATT, PROGRAMMING LANGUAGES: DESIGN & IMPLEMENTATION 19 (2d ed. 1982).

9. *See, e.g.*, the line of cases including Gottschalk v. Benson, 409 U.S. 63 (1972), and

computer innovations were consistent with the continuum that exists for computer innovations. The computer science term for this continuum is "virtual machine."[10] An extended definition of a virtual machine will be discussed in Part III B. For now, a virtual machine may be defined as an innovation that appears to its user as a machine in itself.[11] By modeling all levels of computers as virtual machines, intellectual property law for computers will become more consistent and thus more predictable.

II. A HISTORY OF COMPUTER INTELLECTUAL PROPERTY LAW

In a sense, almost any machine could be called a computer and any process using the machine would be a program,[12] so the history of intellectual property law is the history of computer intellectual property law. Computers, in the narrower sense of digital electronic information processors, date back less than half a century to ENIAC, built in 1946 at the University of Pennsylvania. For the first year, ENIAC was programmed by manually rewiring a control panel. Then John Von Neumann suggested storing instructions in the machine the same way that data is stored.[13] In this way, software, firmware, and all of their corresponding legal problems were born.

A. AN OVERVIEW OF INTELLECTUAL PROPERTY

Intellectual property is an amalgam of legal principles intended to protect the scope of human creativity. This goal is in constant tension with concerns that such protection will stifle future creativity or that it

---

ending with Diamond v. Diehr, 450 U.S. 175 (1984); Apple Computer, Inc. v. Formula Int'l, Inc., 725 F.2d 521 (9th Cir. 1984). *See also supra* notes 3 and 5.

10. *See* Appendix D.

11. *Cf.* "The virtual machine is the computational unit a user confronts when he or she sits down to use [a computer]." Samuelson, *CONTU Revisited: The Case Against Copyright Protection for Computer Programs in Machine Readable Form*, 1984 DUKE L.J. 663, 680.

12. Automated mechanical computers preceded the modern electronic computer by over a century. Charles Babbage had a programmable mechanical calculating machine by 1833. If a computer may be defined as a "device capable of solving problems by accepting data, performing prescribed operations on the data, and supplying the results of these operations," H. HANNEMAN, *supra* note 8, at 3, then any machine, even a simple lever or a pulley is a computer. An even more radical perspective is possible. Edward Fredkin, a former professor of Computer Science at M.I.T., argues that the entire universe is a cellular automation in which energy and matter are composed of information. Wright, *Did the Universe Just Happen?*, ATLANTIC, Apr. 1988, at 29. While one need not believe the universe is a computer to understand this Article, contemplating the possibility may be a useful mental stretching exercise to exorcise rigid preconceptions about computers.

13. H. HANNEMAN, *supra* note 8, at 34.

will be improperly conferred on unworthy efforts. The United States Constitution empowers Congress to encourage progress in science and art by granting innovators exclusive rights to their creations for limited periods of time.[14] These exclusive rights confer enormous economic power. This economic stranglehold could be used to thwart or seriously delay further progress. This is particularly true when the innovation is simply a concept or a universal truth. To balance these concerns, Congress created statutory guidelines for protected subject matter under both the patent[15] and copyright[16] regimes.[17]

1. *The Features and Drawbacks of Patents*

In the United States, patents are used to protect "any new and useful *process, machine*, manufacture, or composition of matter, *or any new and useful improvement thereof* . . . subject to the conditions and requirements of" the Patent Act.[18] There are many different types of patents including patents for plants,[19] patents for designs,[20] patents for drugs,[21] and plain old utility patents.[22] Utility patents protect machines and physical processes,[23] but not processes like methods of doing business.[24] The key requirements are utility, novelty, non-obviousness, and patentable subject matter.[25] An inventor whose invention satisfies

14. *See supra* note 1.

15. Patent Act, 35 U.S.C.A. § 101 (West Supp. 1987) ("Inventions patentable").

16. Copyright Act, 17 U.S.C.A. § 102 (West Supp. 1987) ("Subject matter of copyright: In general").

17. Other intellectual property protections exist such as trademark and trade secret. Trade secret, in particular, is a very useful protection for many computer innovations. It was the first, and may still be the most common, form of software protection. H. HANNEMAN, *supra* note 8, at 7. Trade secret and other forms of protection are not mutually exclusive. *Cf.* Kewanee Oil Co. v. Bicron Corp., 416 U.S. 470 (1974). A thorough discussion of trade secret law relating to computers is beyond the scope of this Article; instead, I shall focus on the more formal means of protection, copyright and patent, which offer exclusive rights for limited terms. Other forms of protection such as trademark are of limited applicability to computer innovations.

18. 35 U.S.C.A. § 101 (West Supp. 1987) (emphasis added).

19. *Id.* §§ 161-64.

20. *Id.* §§ 171-73.

21. *Id.* §§ 155-56. Drugs may receive special patent term extensions. Otherwise, drugs are treated as utility patents.

22. Unless otherwise specified, "patents" in this Article refers to utility patents.

23. Tilghman v. Proctor, 102 U.S. 707 (1880). Tilghman's patent application described "a process for producing free fat acids and solution of glycerine from those fatty and oily bodies of animal and vegetable origin which contain glycerine as their base." *Id.* at 718.

24. *See* Hotel Security Checking Co. v. Lorraine Co., 160 F. 467 (2d Cir. 1908); *but see* Paine, Webber, Jackson & Curtis, Inc. v. Merrill Lynch, Pierce, Fenner & Smith, Inc., 564 F. Supp. 1358 (D. Del. 1983) (Ironically, a method of business is patentable if implemented on a computer).

25. 35 U.S.C.A. §§ 101-03 (West Supp. 1987).

these requirements is entitled to apply for a patent. If the patent office awards a patent, the inventor receives a seventeen year "right to exclude others from making, using, or selling [any embodiments of] the invention throughout the United States . . . ."[26] Congress encourages the inventor by allowing seventeen years to recoup any investment in the research and development leading to the invention.

Patent law is rife with terms of art, starting with the Constitution, which grants to "Inventors the exclusive Rights to their respective . . . Discoveries . . . ."[27] The Patent Act also explicitly includes discoveries,[28] but case law excludes discoveries of anything already occurring in nature and of natural laws.[29] The most important terms of art are "anticipation" and "prior art" because they define novelty. An innovation is anticipated if it is known or used in any of several ways described by statute (e.g., publication more than a year before, public use, or sale), or if it falls within certain statutory bars (e.g., abandonment, claimant not the inventor, etc.).[30] An innovation fails the non-obviousness requirement "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."[31] Section III(C) will explore the problems with the term of art "process."

The seventeen-year exclusive right is a very powerful entitlement. It even precludes identical inventions developed independently of the patented invention.[32] A patent on a useful or popular item can be worth a great deal, but not all inventions are market successes.[33] The decision to apply for a patent is difficult with all but the most sure-fire technologies because the cost of a patent is very high. Since patent applications must be approved by the Patent and Trademark Office (PTO) to have any legal weight, the inventor must engage in an extensive bureaucratic process and pay several fees[34] without any guarantee of success. Moreover, any successful application must distinguish the

26. *Id.* § 154. This is a bit simplified since it ignores the special features of design patents, drug patents, and other kinds of patents, as well as the Sisyphean labor required to get a patent, but it is essentially true for utility patents on computer innovations. The patent may pass to the inventor's assigns and heirs. *See id.* §§ 117, 183, 261.

27. *See supra* note 1.

28. "The term 'invention' means invention or discovery." 35 U.S.C.A. § 100(a) (West Supp. 1987).

29. The irony is that, while the Constitution and the statute explicitly include discoveries, the courts have defined discoveries as unpatentable natural law. *See supra* note 3.

30. This is drastically oversimplified. *See* 35 U.S.C.A. §§ 102, 104 (West Supp. 1987).

31. *Id.* § 103.

32. *Id.* § 271.

33. *Remember Steam Cars and Plastic Teeth?*, ECONOMIST, Sept. 1986, at 71.

34. *See generally* 35 U.S.C.A., chs. 4, 11, 12 (West Supp. 1987).

invention from the prior art. This requires extensive searches by expensive patent attorneys. Once the patent issues, the patent owner must pay maintenance fees.[35] Even after a patent issues to the inventor, competitors may still get the patent declared invalid if they can convince a judge that the invention fails to meet one of the requirements for a patent. Patents are very powerful, but they are no panacea.

2. *The Features and Drawbacks of Copyrights*

Copyright is, of course, no panacea either. It is, however, extremely easy to get. A work is protected in copyright whenever it is visibly marked with a notice.[36] It requires little effort and no lawyers. A copyright need not be registered.[37] Registration is only required when filing suit for infringement.[38] Registering is simple and the registration fee is small.[39] Copyrights last much longer than patents. The duration of a copyright is the lifetime of the creator plus fifty years[40] or, for works for hire, seventy-five years.[41] The subject matter of copyright is "original works of authorship fixed in any tangible medium . . . ."[42] This is very broad and flexible since it encompasses any fixed expression; however, only the expression is protected.

"In no case does copyright protection . . . extend to any idea, *procedure, process*, system, *method of operation*, concept, principle, or discovery, regardless of the form in which it is described, explained, illustrated, or embodied . . . ."[43] The Copyright Act of 1976 was amended in 1980 to define a computer program as "a set of *statements or instructions* to be *used* directly or indirectly in a computer in order *to bring about a certain result.*"[44] While the Copyright Act explicitly covers programs, there are exceptions[45] and ambiguous areas.[46] Unlike

---

35. *Id.* § 41.

36. 17 U.S.C.A. § 401 (West Supp. 1987). The most common copyright notice consists of a copyright symbol, the year copyrighted, and the name of the copyright owner. An example of the most common copyright notice may be found in this volume of the COMPUTER/LAW JOURNAL at page i. The copyright notice for this Article would read: © 1989 *By Richard L. Torczon*

37. *Id.* § 408(a).

38. *Id.* § 411.

39. *Id.* §§ 408, 409, 708.

40. *Id.* § 302(a).

41. *Id.* § 302(c). Works for hire are works produced by an agent of another. This duration also applies to anonymous and pseudonymous works. *Id.*

42. *Id.* § 102(a).

43. *Id.* § 102(b) (emphasis added).

44. *Id.* § 101 (West Supp. 1988) (emphasis added).

45. *Id.* § 117 exempts limited copying by owners of copies of software to make archival copies or to make copies "as an essential step in the utilization of the computer program in conjunction with a machine . . . ." *Id.* § 117(1) (Supp. 1988).

46. In Apple Computer, Inc. v. Formula Int'l, Inc., 725 F.2d 521 (9th Cir. 1984), the

the patent system, copyrights are not exclusive in the sense that any independently created suspect item is not an infringement. This means a copyright plaintiff must prove that the facsimile was most likely copied.

Copyright seems to have enormous advantages for the inventor. Copyright protection effectively lasts forever,[47] and getting a copyright is vastly easier and cheaper than getting a patent. Unfortunately, since programs are designed with specific functions in mind, several programmers facing the same problem may independently develop nearly identical software. None would have the right to exclude the others as long as the software was developed independently. Copyright also impedes software evolution by allowing easy protection for trivial improvements, and by making the protection much longer than the useful life of virtually any software. One approach to the latter problem is to create shorter terms for copyrights on software.[48] This approach has largely been ignored in the United States.

B. THE CURRENT STATUS OF COMPUTER INTELLECTUAL PROPERTY LAW

American computer intellectual property follows four divergent lines. First, hardware is always patentable subject matter. Second, algorithms are not patentable, with some exceptions. Third, software is copyrightable subject matter. Finally, when all else fails, Congress will create *sui generis* protection for some things.

1. *Hardware May Be Patented*

"Hardware consists of tangible objects—integrated circuits, printed

Ninth Circuit upheld an Apple copyright on parts of an operating system (typically software) built into memory chips (hardware). This is copyright protection for firmware.

47. In the fast-moving software world, twenty years is a very long time to exist without significant changes. UNIX, a widely used operating system that is one of the older programs still in wide use, has been drastically revised and improved in the last two decades. Some UNIX user manuals have been copyrighted. UNIX is a trademark of Bell Laboratories. B. KERNIGHAN & D. RITCHIE, THE C PROGRAMMING LANGUAGE ix (1978). UNIX is written primarily in C, but neither C nor UNIX commands like *shell*, *nroff*, or *echo* are trademarks. At least one feature, the *set-userid bit*, is patented by Western Electric. T. PLUM, UNIX OPERATING SYSTEM WORKSHOP 0.1 (Sept. 1982). Both Western Electric and Bell Labs are subsidiaries of American Telephone and Telegraph Company.

48. Japan's Ministry of International Trade and Industry (MITI) considered a plan for software copyrights that required registration, granted copyright protection for fifteen years, and made software subject to compulsory licensing. Faced with the trend toward ordinary copyrights for programs in most developed countries, MITI has currently shelved its plan. H. HANNEMAN, *supra* note 8, at 12. *Cf.* Samuelson, *CONTU Revisited*, *supra* note 11; and Samuelson, *Creating a New Kind of Intellectual Property: Applying the Lessons of the Chip Law to Computer Programs*, 70 MINN. L. REV. 471 (1985) (advocating *sui generis* protection for software).

circuit boards, cables, power supplies, memories, card readers, line printers, and terminals—rather than abstract ideas, algorithms, or instructions."[49] This is undoubtedly the stuff of patent.[50] Hardware is, of course, still subject to the requirements of novelty, utility, and non-obviousness. Hardware is the easy case. Any deviation from a tangible physical device (as in the case of firmware) throws the whole patent regime into doubt.

2. *The* Benson-Flook-Diehr *Line of Cases on Patentability of Algorithms*

As already noted, ideas and natural laws are not patentable.[51] The Court of Customs and Patent Appeals (CCPA) was scrupulous about rejecting patents for methods that bordered on ideas. Applications were rejected as attempts to patent mental steps[52] or mathematical calculations.[53] The PTO extended this reasoning to computer programs.[54] The PTO even drew distinctions between analog and digital computers.[55] This set the groundwork for the Supreme Court to consider the patentability of algorithms.

An algorithm is simply "a step-by-step procedure for solving a problem or accomplishing some end . . . ."[56] The Court of Customs and Pat-

---

49. A. TANENBAUM, *supra* note 5, at 10.

50. *See* 35 U.S.C.A. § 101 (West Supp. 1987) (machines may be patented). Also, Webster's Dictionary, defines "machine," at subsense f, as "a mechanically, electrically, or electronically operated device for performing a task . . . ." WEBSTER'S NEW COLLEGIATE DICTIONARY 682 (1979). To my knowledge, the patentability of hardware was never in doubt.

51. *See* this note, *supra* section II(A)(1).

52. *E.g., In re* Abrams, 188 F.2d 165 (C.C.P.A. 1951). In *Abrams*, a method of prospecting for petroleum deposits by detecting gas anomalies in a vicinity was rejected as being purely mental in character. Although machines were used in the method, the novel steps were all mental, so the method was not patentable.

53. Application of Shao Wen Yuan, 188 F.2d 377, 383 (C.C.P.A. 1951) ("sole novelty in the claim . . . resides in the method of mathematical computation by which . . . the airfoil is determined.").

54. H. HANNEMAN, *supra* note 8, at 37 (citing in ex parte King and Barton, 146 U.S. P.Q. 590 (Pat. Off. Bd. App. 1964)).

55. "In the second form of the apparatus rejection, the Examiner stated that claim 10 read on the analog computer for which the applicants were entitled to patent coverage and the properly programmed general-purpose digital computer for which applicants were not entitled to coverage . . . ." H. HANNEMAN, *supra* note 8, at 41 (discussing the background of Application of Prater, 415 F.2d 1378 (C.C.P.A. 1968)).

56. WEBSTER'S NEW COLLEGIATE DICTIONARY, *supra* note 50, at 28; "An algorithm is an unambiguous specification of a conditional sequence of steps or operations for solving a class of problems." Newell, *Response: The Models are Broken, The Models are Broken*, 47 PITT. L. REV. 1023 (1986). To anyone but a patent lawyer or a patent examiner, this sounds suspiciously like a process; however, Allen Newell, the U.A. and Helen Whitaker, University Professor of Computer Science at Carnegie-Mellon University, argue that any

ent Appeals decided in 1970 to allow a patent on an algorithm for converting binary-encoded decimal numerals (BCDs) into binary numerals.[57] The CCPA argued that "all that is necessary to make a process a statutory 'process' within 35 U.S.C. section 101 is that it be in the technological arts, while a 'standard of reasonableness' should be applied to evaluate the interpretation of claims."[58] It went on to conclude that "a process having no practical value other than enhancing the internal operation of those machines is . . . likewise in the technological or useful arts."[59] However, on appeal, the Supreme Court unanimously rejected the patent application.[60] "It is conceded that one may not patent an idea. But in practical effect that would be the result if the formula for converting BCD numerals to pure binary numerals were patented in this case."[61] The Supreme Court stressed its concern that granting a patent here would "wholly preempt the mathematical formula."[62] The Court also insisted that its holding did not freeze patents to old technologies and that process patents, including those for analog computers, were still possible.[63]

The Supreme Court next addressed the issue in *Parker v. Flook*.[64] The patent involved an automated system for measuring physical variables in a catalytic conversion process and constantly recalculating what

---

"attempt to erect a patent system for algorithms that tries to distinguish algorithms as one sort of thing and mental steps as another, will ultimately end up in a quagmire." *Id.* at 1025. Newell's concern, predicated in part on cognitive psychology, is invalid unless it applies to any method or process. An algorithm or recipe that is physically implemented is not a mere mental step. A procedure for vectorizing and executing on an SIMD machine outer loops in the presence of recurrent inner loops is no more a set of "mental steps" than is a method for separating fatty acids and glycerine.

57. Application of Benson, 441 F.2d 682 (C.C.P.A. 1971), *rev'd*, 409 U.S. 63 (1972).

58. H. HANNEMAN, *supra* note 8, at 49 (discussing Application of Benson).

59. Application of Benson, 441 F.2d at 688.

60. Gottschalk v. Benson, 409 U.S. 63 (1972).

61. *Id.* at 71.

62. *Id.* at 73. However, the C.C.P.A. had found that the claim

> when reasonably interpreted, did not cover the process when implemented by the human mind, in particular because the claim referred to a specific, hardware apparatus, the "reentrant shift register," and could be carried out with no intervention by a human being once the apparatus was set up . . . . [W]hen reasonably interpreted, each of the operational steps of the claim were prescribed so that no human judgment or decision was required. The Court added that only in the manual performance would [the algorithm] require the operator to think and then only to the extent necessary to assure that he was doing what the claim told him to do.

H. HANNEMAN, *supra* note 8, at 49 (discussing Application of Benson, 441 F.2d 682). Faced with precisely the same algorithm, the West German Federal Patent Court granted a patent. *Id.* at 175.

63. *Benson*, 409 U.S. at 71.

64. 437 U.S. 584 (1978). The Court avoided the issue in Dann v. Johnston, 425 U.S. 219 (1976), by rejecting the patent for obviousness.

kind of readings would be abnormal and hence dangerous. The CCPA had approved the application because the claim included "post-solution activity, a step in which the solution is applied to a control system" and for that reason, use of the mathematical formula in the algorithm by itself was not preempted.[65] The Supreme Court reversed,[66] arguing that the mathematical formula used was not new, and that granting a patent would preempt an idea.

> The notion that post-solution activity, no matter how conventional or obvious in itself, can transform an unpatentable principle into a patentable process exalts form over substance. A competent draftsman could attach some form of post-solution activity to almost any mathematical formula; the Pythagorean theorem would not have been patentable, or partially patentable, because a patent application contained a final step indicating that the formula, when solved, could be usefully applied to existing surveying techniques.[67]

Moreover, the Supreme Court made "process" a term of art by explicitly excluding algorithms.[68] For the Supreme Court, a novel use of a mathematical formula in a patent could not support a patent.[69]

The dissent argued that one unpatentable step should not void the entire patent. Writing for the dissent, Justice Stewart noted:

> that thousands of processes and combinations have been patented that contained one or more steps or elements that themselves would have been unpatentable subject matter. . . . The Court today says it does not turn its back on these well-settled precedents, . . . but it strikes what seems to me an equally damaging blow at basic principles of patent law by importing into its inquiry [about subject matter] . . . the criteria of novelty and inventiveness.[70]

By 1980, it had become clear that the *Benson-Flook* rule was not

---

65. Application of Flook, 559 F.2d 31 (C.C.P.A. 1977), *rev'd*, 437 U.S. 584 (1978).

66. The decision was 6-3.

67. *Flook*, 437 U.S. at 590. The majority missed the point. The Pythagorean theorem would not be patented, a process using the theorem would be patented. By the Court's reasoning, Tilghman's fat acid-glycerine separation process should not have been patented because it includes a novel use of the venerable concept of pressure cooking. *See supra* note 24.

68. *Flook*, 437 U.S. at 588. The Court stated:

> It is true, as respondent argues, that his method is a "process" in the ordinary sense of the word. But that was also true of the algorithm . . . involved in *Gottschalk v. Benson*. The holding that the discovery of that method could not be patented as a "process" forecloses a purely literal reading of § 101 [statutory subject matter].

Id.

69. "Even though a phenomenon of nature or mathematical formula may be well known, an inventive application of the principle may be patented. Conversely, the discovery of such a phenomenon cannot support a patent unless there is some other inventive concept in its application." *Id.* at 594.

70. *Id.* at 599-600.

working. With computer controls used throughout industry, manufacturers could not afford to forgo either the patent protection or the programs since, in the current state of the art, software is necessary for flexibility,[71] and flexibility enhances marketability, thus increasing the need for patent protection. This continuum came to a head in *Matter of Application of Bradley*[72] and in *Application of Diehr*.[73]

Bradley's patent application claimed a firmware module that directed data transfers between scratchpad registers and the main memory in Honeywell's Series 60 Level 64 computer.[74] The CCPA held that the claim covered "a combination of hardware elements, one of which happens to be a portion of the computer's control store microprogrammed in a particular manner."[75] The Diehr patent application claimed rights to an automated rubber molding process which used a computer controller programmed to calculate the cure time for rubber seals using a well-known equation.[76] The CCPA applied a two-part test it had developed.[77] It held that the "recitation of the equation is not separable from the process in which it is used; it is intimately involved in the process, but the claims are not to the equation."[78] *Matter of Application of Bradley* and *Application of Diehr* went up to the Supreme Court together.[79]

This hearing triggered *amicus curiae* briefs from diverse industries ranging from computer manufactures to oil companies.[80] Curiously, two leading computer trade groups did not submit briefs: one because it expected trade secret doctrine would prove a more fruitful avenue of

---

71. *Cf. supra* text accompanying note 9.

72. 600 F.2d 807 (C.C.P.A. 1979), *aff'd*, 450 U.S. 381 (1981).

73. 602 F.2d 982 (C.C.P.A. 1979), *aff'd*, 450 U.S. 175 (1981).

74. *Patentability of Stored Programs Will Be Judged By Supreme Court*, ELECTRONIC NEWS, Oct. 6, 1980, at 40 [hereinafter *Patentability*].

75. *Application of Bradley*, 600 F.2d at 812. This classification of microprogramming as hardware is a bit strained. A simpler solution would be to allow patents on microprograms, but that would have been contrary to *Benson* and *Flook*.

76. *Patentability*, *supra* note 74. The equation is Arrhenius's Equation. The patent beneficiary was National Mogul Corp. of Detroit.

77. *In re* Freeman, 573 F.2d 1237, 1245 (1978). The court, applying its interpretation of *Benson*, stated:

> First, it must be determined whether the claim directly or indirectly recites an "algorithm" in the *Benson* sense of that term, for a claim which fails even to recite an algorithm clearly cannot wholly preempt an algorithm. Second, the claim must be further analyzed to ascertain whether in its entirety it wholly preempts that algorithm.

Id.

78. Application of Diehr, 602 F.2d 982, 988 (C.C.P.A. 1979), *aff'd*, 450 U.S. 175 (1981).

79. Diamond v. Bradley, 450 U.S. 381 (1981), was decided six days after *Diehr*. The decision was split (4-4, Chief Justice Burger took no part) so the C.C.P.A. opinion was upheld without opinion.

80. *Patentability*, *supra* note 74.

protection, the other because it did not think the issues were clear cut enough for the Supreme Court.[81] The American Patent Law Association submitted briefs arguing that the PTO had over-zealously converted the inventions into programs, which were unpatentable. In this contentious atmosphere, the Supreme Court decided *Diamond v. Diehr*.[82]

Diehr made algorithms patentable subject matter "when a claim containing [an algorithm] implements or applies that formula in a structure or process which, when considered as a whole, is performing a function which the patent laws were designed to protect . . . ."[83] Patentable subject matter does not extend to the algorithm in the abstract but only to its implementations. The invention must still satisfy the separate requirements of novelty, utility, and non-obviousness. Since a program is just a computer representation of an algorithm,[84] under a broad reading, programs should always be patentable.[85]

The dissent clung to the *Benson* decision that algorithms were not patentable. Writing for the dissent, Justice Stevens even argued that the scope of *Benson* should be extended. "[T]he term 'algorithm' as used in this case, and in *Benson* and *Flook*, is synonymous with the term 'computer program.' "[86] This would make software unpatentable.

3. *Software is Copyrighted*

In the beginning, software was not protected. Hardware manufacturers like International Business Machines Corporation ("IBM") gave it away to encourage sales. Manufacturers sponsored user's groups whose members shared software. When programmers wanted to avoid sharing software, they simply kept the program a secret.[87] The PTO first faced the question of software patentability in 1964. Programs were deemed "creations in the area of thought" and thus rejected as patentable subject matter.[88] The PTO was also worried about the administrative burden of evaluating a program—a responsibility PTO officers feared would be difficult, time-consuming, and beyond their expertise.[89] The

81. The two groups are A.D.A.P.S.O., the Computer Software and Services Industry Association, and C.B.E.M.A., the Computer and Business Equipment Manufacturer's Association, respectively. *Patentability*, *supra* note 74.

82. 450 U.S. 175 (1984) (a 5-4 decision).

83. *Id.* at 192.

84. A. TANENBAUM, *supra* note 5, at 10.

85. *See* Appendix B.

86. *Diehr*, 450 U.S. at 219.

87. H. HANNEMAN, *supra* note 8, at 36.

88. *Id.* at 37.

89. *Id. See also "To Promote the Progress of . . . Useful Arts" In an Age of Exploding Technology*, REPORT OF THE PRESIDENT'S COMMISSION ON THE PATENT SYSTEM 13 (1966):

The Patent Office now cannot examine applications for programs because of a

Benson Court noted this trend in dicta, and recommended congressional action.[90]

Congress responded by creating the National Commission on New Technological Uses of Copyrighted Works [hereinafter CONTU].[91] The CONTU Report ambitiously covered both computer programs and photocopying. CONTU was concerned about two aspects of computers: first, computers had become less expensive, more powerful, and more common throughout society; and second, computers had become sufficiently standardized that many programs were readily transferable between different machines.[92] The authors feared that the ease of copying and the high cost of software development would make software production uneconomical outside of subsidized environments.[93]

To protect software from piracy,[94] CONTU recommended the use of copyrights. The report noted that the Copyright Office already registered computer programs, and that studies for the United Kingdom and for the World Intellectual Property Organization had recommended protections similar to American copyright protections.[95] The report went on to compare existing American intellectual property protections and concluded "copyright has the smallest negative impact."[96]

CONTU recommended Congress rewrite section 117 of the Copyright Act to allow owners to make archival copies, and to exclude loading of a program into a computer from the definition of piracy.[97] The

---

lack of a classification technique and the requisite search files. Even if these were available, reliable searches would not be feasible or economic because of the tremendous volume of prior art being generated. Without this search, the patenting of programs would be tantamount to mere registration and the presumption of validity would be all but non-existent.

It is noted that the creation of programs has undergone substantial and satisfactory growth in the absence of patent protection and that copyright protection for programs is presently available.

90. "If these programs are to be patentable, considerable problems are raised which only committees of Congress can imagine . . . . The technological problems tendered in the many briefs [including 14 amicus briefs] before us indicate to us that considered action by Congress is needed." Gottschalk v. Benson, 409 U.S. 63, 73 (1972).

91. NATIONAL COMMISSION ON NEW TECHNOLOGICAL USES OF COPYRIGHTED WORKS, FINAL REPORT 1 (1978) [hereinafter CONTU REPORT].

92. *Id.* at 10.

93. *Id.* at 11. Subsidized environments might include government, academic, and hardware manufacturer's marketing projects.

94. Piracy colloquially means criminal copyright infringement. *See* 17 U.S.C.A. § 506(a) (West Supp. 1987).

95. CONTU REPORT, *supra* note 91, at 11 nn.428-44. The report also noted that a Canadian study came to "the opposite conclusion," and that an Australian report on "reprographic reproduction" considered computer issues outside its scope. *Id.* at 11-12.

96. That is, the fewest disadvantages. *See id.* at 18 & table 1 at 19 (reproduced herein as Appendix C). This is hardly a ringing endorsement. The authors were probably most concerned about copyright's inability to protect the underlying process. *Id.* at 20-21.

97. *Id.* at 12.

report also recommended an amendment to section 101 to define a " 'computer program' [as] a set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result."[98]

The CONTU report had a concurrence and two dissents.[99] In his concurrence, Commissioner Nimmer worried that the majority's recommendation was too open-ended and thus stretched the meaning of "authors" and "writings." This could be a constitutional problem. He suggested a possible demarcation between programs that produce potentially copyrightable material (such as word processors, databases, and games) and those that do not (such as catalytic-conversion controlling software).[100]

Commissioner Hersey strongly dissented from the majority position. Basing his arguments on M.I.T. Professor Joseph Weizenbaum's book, *Computer Power and Human Reason*, Hersey argued that computer protection should end.[101] Congress chose to ignore Hersey's apocalyptic vision. It adopted CONTU's recommendations in full.

The new copyright law paved the way for greater copyright protection in computer intellectual property law. One consequence is the

---

98. *Id.* at 12. This definition, with its focus on results, seems suspiciously like a definition of a useful process.

99. *Id.* 26-38. Commissioner Karpatkin's dissent is a lukewarm support for Commissioner Hersey's dissent, so it will not be discussed.

100. *Id.* at 26-27. This distinction is troublesome since it would require distinguishing a database used to record recipes from a database used to monitor catalytic conversion. Such a distinction would necessarily be artificial. Nimmer analogizes a program to a sound recording. *Id.* at 27. Both, he notes, tell a machine how to do something. The problem here is that even if a record's groove pattern were patentable, the value is in the sound it makes every time it plays. Most programs do not produce the same output every time; generally, a program that did would not be flexible enough to be of much value. If a program always produced the same graphic design, that design would be copyrightable with the property interest vested in the programmer. *See* Samuelson, *Allocating Ownership Rights in Computer-Generated Works*, 47 U. PITT. L. REV. 1185 (1986) (generally, rights in computer-generalized works should go to the user). If rights went to the user, Nimmer's analogy would hold, but only because in each case the thing of greatest value is the output, not the controlling method.

101. Commissioner Hersey wrote:

> Congress should weigh most carefully the heavy responsibility of breaking with tradition and enabling, by law of the land, for the first time ever, copyright protection for communication, not with our fellow human beings, but with machines—thus equating machines with human beings as intended recipients of the distribution that copyright was designed to foster.
>
> Surely it is especially vital, in a time of hurtling and insatiable technology, that the nation's laws reflect, whenever possible, a distinction between the realm and responsibility of human beings and the realm and responsibility attributed to machines.

CONTU REPORT, *supra* note 91, at 37 (discussing J. WEIZENBAUM, COMPUTER POWER AND HUMAN REASON (1976)).

Semiconductor Chip Protection Act of 1984.[102] Congress was besieged by chip manufacturers clamoring for easy protection for the masks used in manufacturing chips.[103] Congress complied by creating a *sui generis* form of quasi-copyright. Also in 1984, the Ninth Circuit decided that microcode was protected by copyright law.[104] At least one commentator has called for an additional *sui generis* form of protection like the Semiconductor Chip Protection Act for microcode.[105]

Despite two decades of review and discussion by Congress, courts, commissions, and academicians, computer intellectual protection law remains unsettled. On one hand, the *Benson-Diehr* line of cases covered the extremes but currently seems to encourage patent protection for at least some algorithms, and the patent office will grant a patent for an algorithm standing on its own. On the other hand, Congress codified CONTU's recommendations that software be the proper subject of copyright. Congress also carved out a *sui generis* quasi-copyright for "semiconductor chip product[s]." Rather than settling on any coherent set of solutions, the approaches seem to be diverging—in some cases on precisely the same subject matter. Lest we Americans feel alone in our confusion, Japan, France, West Germany, and virtually every other major First World nation has waffled or flip-flopped on these very problems in the last two decades.[106]

C. PROBLEMS WITH THE CURRENT APPROACH

As the forgoing discussion demonstrates, computer intellectual property is in disarray. No single regimen for protection exists for any computer innovation. Trade secret can be used to protect almost all computer innovation. Patent and copyright protect potentially overlapping areas of innovation. The duplication between patent and copyright on one hand, and trade secret on the other is the product of a choice between an open, formal system and a guarded, informal system. The duplication between copyright and patent is less benign. Since they are

---

102. Chip is a colloquial term for integrated circuits.

103. Semiconductor Chip Protection Act of 1984, Pub. L. No. 98-620, tit. III, § 302, 98 Stat. 3347-55 (codified as 17 U.S.C.A. § 901-14 (West Supp. 1988)). Samuelson, *Creating a New Kind of Intellectual Property*, 70 MINN. L. REV. 471, 476-77 (1985). Masks are stencils used to manufacture chips by means of etching and depositing on semiconductor wafers. *See generally* Kastenmeier & Remington, *The Semiconductor Chip Protection Act of 1984: A Swamp or Firm Ground?*, 70 MINN. L. REV. 417 (1985).

104. Apple Computer, Inc. v. Formula Int'l, Inc., 725 F.2d 521 (9th Cir. 1984), *aff'g* 562 F. Supp. 775 (C.D. Cal. 1983) (preliminary injunction was properly issued to prevent Formula from copying copyrighted object code fixed in integrated circuits as firmware).

105. Harris, *Legal Protection for Microcode and Beyond: A Discussion of the Applicability of the Semiconductor Chip Protection Act and the Copyright Laws to Microcode*, 6 COMPUTER/L.J. 187, 212 (1985).

106. *See generally* H. HANNEMAN, *supra* note 8.

both open, formal, constitutional means of protection, their duplication of protection raises serious constitutional and policy questions.[107] Confusion between the systems increases transaction costs in the form of unnecessary legal fees and litigation. It also allows crafty applicants to exploit the confusion by circumventing policy goals.[108]

1. *The Treatments are Inconsistent*

The scopes and the natures of patent and copyright are very different. Copyright is intended to protect broadly defined artistic expression. Patent is intended to protect narrowly defined useful invention or design. Computers are useful inventions, but only parts of a computer are treated as such. Hardware is treated as invention; software is generally treated as artistic expression regardless of its utility. Yet, exceptions exist within that general rule,[109] so even software is treated inconsistently. This inconsistency is bad since it defeats the policy for having two separate protection schemes.

2. *The Current Treatment Does Not Satisfactorily Handle Gray Areas*

What appears to be a computer to the user is a virtual machine: a continuum of hardware and software. This continuum includes intermediate elements like firmware,[110] microprograms,[111] and object code[112] that are legally neither fish nor fowl. Since it is a continuum, an infinite number of intermediate elements could exist. With the cur-

---

107. Kline, *Requiring an Election of Protection for Patentable/Copyrightable Computer Programs*, 6 COMPUTER/L.J. 607 (1986).

108. *Id.* at 638. Kline argues that copyright is improperly used to extend patent protection and that this is at odds with the constitutional mandate.

109. *E.g.*, U.S. Patent 4,710,872 (December 1987) (A method for vectorizing and executing on an SIMD [single-instruction, multiple-data path] machine outer loops in the presence of recurrent inner loops). This is indisputably a patent on an algorithm.

110. Apple Computer, Inc. v. Formula Int'l, Inc., 725 F.2d 521 (9th Cir. 1984).

111. *See supra* notes 6-8 and accompanying text.

112. *See, e.g.*, Note, *Copyrighting Object Code: Applying Old Legal Tools to New Technologies*, 4 COMPUTER/L.J. 421, 433-34 (1983):

> The courts and commentators fail to understand how the personal computer industry functions. They assume that the issue is whether code in a program communicates with a human audience, either directly or indirectly. They have no problems with programs written in source code, because source code programming is understood by many people. However, some have decided that object code fails this test.

Source code is a program written by a programmer in a high-level language, i.e., a mnemonic language. Object code is source code that has been converted by a program called a compiler into machine-readable instructions. Object code needs to be "linked" by a program called a linker to other bits of object code and to certain parts of the computer's memory before it can be used. *Cf.* A. TANENBAUM, *supra* note 5, at 371.

rent *ad hoc* approach to this continuum, each element must be separately litigated, or, at best, arbitrarily assigned to an existing category.

3. *The Current Treatment is Not Flexible Enough to Handle Likely Innovations*

Some likely innovations will probably work well within the current regime. For instance, computer-generated artwork and programs will likely be handled quite adequately by existing copyright law.[113] Others will be more difficult to classify. John Von Neumann created the notion of software as a useful and flexible amplification of hardware. Within one year of its creation, software was a firmly established part of computer innovation. Innovators may be on the brink of currently unimaginable innovations that will defy classification by the current regime. In the more mundane world of immediately likely innovations, any number of intermediate devices are possible, and the upper end of software is unlimited: it may eventually include even program-writing programs.[114]

The current morass in computer intellectual law is the consequence of a mistaken concept of what constitutes a computer.[115] This regime envisions a bright line between software and hardware; but it stumbles on all of the shades of gray in between. The result is a confusing and inflexible protection system that lends itself to inadvertent misuse and intentional abuse. In his dissent to the CONTU Report, Commissioner Hersey wrote:

> In the early stages of its development, the basic ideas and methods to be contained in a computer program are set down in written forms, and these will presumably be copyrightable with no change in the . . . [Copyright Act of 1976]. But the program itself, in its mature and usable form, is a machine-control element, a mechanical device, which on constitutional grounds and for reasons of social policy ought not be copyrighted.[116]

---

113. M. WESSEL, FREEDOM'S EDGE: THE COMPUTER THREAT TO SOCIETY 109-12 (1974) raises some of these concerns. They are brilliantly answered in Samuelson, *supra* note 100 (generally, rights in computer-generated works should go to the user).

114. Such programs are expected to be necessary to create the enormous control programs envisioned in the Strategic Defense Initiative. Some commentators argue that program-writing programs are just "higher-level programming languages." *See* Pamas, *Software Aspects of Strategic Defense Systems*, 73 AM. SCIENTIST 432, 438 (1985), *reprinted in Strategic Defense Initiative: Hearings Before the Subcomm. on Strategic and Theater Nuclear Forces of the Senate Comm. on Armed Services*, 99th Cong., 1st Sess. 330, 336 (1985).

115. "The *Benson* decision is not very well reasoned either, leaving much room for speculations about its meaning . . . . The difficulties that non-scientists have in understanding computer technology are regarded as a source of the *Benson* Court's confusion." H. HANNEMAN, *supra* note 8, at 54.

116. CONTU REPORT, *supra* note 91, at 27.

If the program is a machine element, the intermediate processes and devices must be as well.

III. A MODEL FOR COMPUTERS AND COMPUTER PROGRAMS

So far, this Article has built on general, hazy notions of what constitutes a computer while hinting that a useful model exists. The model contemplated is the virtual machine. The virtual machine model integrates hardware, software, and the various intermediate elements into a single stratified model composed of machines and languages. In order to better understand the power of, and need for, this model, this Article offers some history and definitions.

A. AN OVERVIEW OF MODELS FOR COMPUTERS

As noted earlier,[117] almost any machine could be considered a computer. The mathematical principles underlying computers are so powerful that a respected scientist can plausibly argue that the universe is a computer.[118] A definition of computers that broad would render the term useless for legal analysis since, by encompassing everything, it distinguishes nothing. Instead, this Article traces the history of computer elements to develop the concept.

1. *Computers Are Machines*

We are comfortable with the idea that computers are machines. Common experience informs us that a computer is "a programmable electronic device that can store, retrieve, and process data . . . ."[119] For the last four decades that definition has been pertinent, but the idea of computing machines, even automated ones, is very old. A cursory examination of the history of computers reveals that today's electronic computers are just the latest in a venerable series of computing machines.

a. From Babbage to Cray:

Suppose you were an astronomer studying the sun. You might want to use a computer to record the position of the sun at various times of the year in such a way that the information is easily retrievable the following year for comparison. You might also program it to produce certain output when the sun does various things. If your available technology were limited by the fact that you lived in a Bronze Age society, you

117. *See supra* note 12.

118. Wright, *supra* note 12.

119. WEBSTER'S NEW COLLEGIATE DICTIONARY, *supra* note 50, at 230.

might use a configuration of stones on a plain.[120] Although the technology is crude and the output ephemeral, such a configuration would in principle be an automatic analog computer.

While your solar-powered automatic computer might be the first computer that automatically processes data, the use of stones for storage and retrieval was prior art by about 2000 B.C., when the first abaci were in common use.[121] The abacus was refined for the next four millennia until 1622 when English mathematician William Oughtred invented slide rules.[122] From there the pace quickened. In 1642, Blaise Pascal, at the age of nineteen, invented an adding machine to help his father, a tax collector, calculate taxes.[123] Within thirty years, Gottfried William von Leibnitz conceived a machine that could multiply, as well as add, numbers. By 1694 the Leibnitz's machine was used albeit somewhat unsuccessfully to calculate logarithms.[124] Analytical engines were such a rage among natural philosophers that Jonathan Swift parodied them in his *Gulliver's Travels*.[125]

The next advance came when Charles Babbage, an English mathematician and engineer, invented his "difference engine" around 1820. The machine weighed two tons, took four years to build, and accurately calculated several transcendental functions out to six decimal places.[126] Babbage next attempted an "analytical engine" capable of solving any arithmetic problem. The machine would solve them by linking together the different operations involved.[127] Such a machine would have to store "all the variables . . . operated upon, as well as those quantities which have arisen from the result of other operations" and would run a "mill into which the quantities about to be operated upon are always brought."[128] Babbage died in 1871 having exhausted a government grant, his personal fortune, and the limits of the technology of his era.[129]

---

120. This describes Stonehenge, a circular array of stones on Salisbury Plain in England. One theory about Stonehenge proposes it was used to calculate the occurrence of solar eclipses. *See* THE WORLD OF THE COMPUTER 20 (J. Diebold ed. 1973) [hereinafter Diebold].

121. *Id.* at 21.

122. *Id.*

123. Neither of my sources of this information reveal the speed or accuracy with which returns were processed. *Id.*; H. KÜNG, DOES GOD EXIST? 43-44 (E. Quinn trans. 1980).

124. Diebold, *supra* note 120, at 21.

125. SWIFT, GULLIVER'S TRAVELS AND OTHER WRITINGS 181-83 (Bantam Classic ed. 1981) (a drawing of the machine appears on p. 182).

126. The transcendental functions computed included logarithms, sine, and cosine. Diebold, *supra* note 120, at 23. *See also* P. MORRISON & E. MORRISON, CHARLES BAGGAGE AND HIS CALCULATING ENGINE (1961).

127. Diebold, *supra* note 120, at 23.

128. Charles Babbage, *Life of a Philosopher*, in Diebold, *supra* note 120, at 30.

129. Diebold, *supra* note 120, at 23.

Computers were largely ignored until World War II created a demand for computational power and speed. Babbage's work was continued in 1925 when an M.I.T. group under Vannevar Bush built an electrically powered analog calculator. A more accurate model was completed in 1942, but was classified because it was used to calculate artillery firing tables. Howard Aiken began work on an electric version of Babbage's analytical engine in 1939. He became interested while looking for a way to simplify the calculations necessary for his doctoral thesis at Harvard. He approached the International Business Machines Corporation and an industry was born. All of this happened before the first electronic computer was ever built.[130]

The first electronic computer, ENIAC, was developed for the Aberdeen Proving Ground, an Army artillery facility. Two of its designers, J. Presper Eckert and John Mauchly, went on to build the first commercial stored-program computer for Sperry-Rand in 1951.[131] Since then, computer design and computer science have thrived. Leviathan computer companies battle for preeminence in pursuit of "supercomputers" like Cray Research's Cray-3.[132] These computational behemoths use new architectures[133] and exotic materials,[134] but they still follow Babbage's basic design of some memory and at least one processor for the data.

b. Notions of hardware:

As the previous discussion illustrates, computer hardware is not startingly exotic, much less without precedent. Despite the druidic priesthood of data processing employees who guard the mysteries of computers from the uninitiate with forbidding terms like "supercomputer," computers are just machines. Like cars, computers may require a little pampering and cost a small fortune, but they are still just machines.

2. *Computers are Mathematical Models*

Part of the mystique of computers is their relationship with mathematics. The early innovators were mathematicians, scientists, or engineers. This is not uncommon for any innovation, but these people developed computers to help *do* math. Computer scientists use terms like algorithm for plain old step-by-step instructions. When they talk

130. Bernstein, *The Analytical Engine*, in Diebold, *supra* note 120, at 35-43.

131. *Id.* at 42-43.

132. Elmer-DeWitt, *Fast and Smart*, TIME, Mar. 28, 1988, at 54-57.

133. Architectures such as several processors instead of Babbage's single "mill are used."

134. Materials like gallium arsenide, gold, (and, possibly in the near future, superconducting ceramics) are used.

about machines they mean mathematical objects. All of this seems very daunting, but in the abstract it is not all that different or difficult.

a. Jacquard, Turing, and Von Neumann:

Around 1800, the weaving industry desperately needed a cheap, accurate, and flexible method of controlling the enormous number of needles involved in industrial weaving. Joseph Jacquard anticipated computer punch cards by developing a method of punching holes in cards corresponding to the needles to be activated.[135] The method was a success and became widely used throughout industry. It also was one of Babbage's inspirations. The American statistician Herman Hollerith used similar cards to compile data from the 1890 census. Hollerith was so impressed by his success that he decided to commercialize the method. His corporation became one of the companies that merged to form IBM.[136] Later, John Von Neumann applied essentially the same idea to electronic computers.[137] This flexible means of controlling hardware was dubbed software.

While Babbage and his successors sought a machine that could solve all arithmetic problems, mathematicians sought an algorithm for determining the truth or falsity of any mathematical proposition. The effort hit a dead-end in 1931 when Kurt Gödel proved that such an algorithm could not exist.[138] The proof essentially shows that there are no effective procedures to compute some functions.

The generally accepted model for an effective procedure for computing computable functions was described by Alan Turing in 1935.[139] Turing's model, known as a Turing machine, is purely a mathematical model. It hypothesizes a tape, infinitely long in one direction, and divided into cells containing symbols from a finite set of symbols. A control device with a reading and writing head scans the tape. The control device has a finite number of states in which it can be and a set of rules for moving. This machine is the foundation of modern computer science.

While a Turing machine is physically impossible,[140] it theoretically meets Babbage's requirements for a computing machine. In fact, since

135. Diebold, *supra* note 120, at 23.

136. *Id.*

137. *See* H. HANNEMAN, *supra* note 8, at 34.

138. J. HOPCROFT & J. ULLMAN, INTRODUCTION TO AUTOMATA THEORY, LANGUAGES, AND COMPUTATION 147 (1979); Newell, *supra* note 56, at 1024-25.

139. Turing, *On Computable Numbers with an Application to the Entscheidung-problem*, 2 PROC. LONDON MATH. SOC'Y 42 (1936) (a correction in 43 PROC. LONDON MATH. SOC'Y 544-46). Similar models were simultaneously developed by S.C. Kleene, A. Church, and E. Post.

140. For one thing, one could not build an infinitely long tape.

the Turing machine can model any effective procedure whether physically possible or not, it can be mathematically manipulated to model any physical computer. This may seem disturbing at first, but scientists and engineers use mathematics to model physical phenomena all the time. Consequently, a Turing machine can model any program.

b. Notions of software:

The fact that a Turing machine can model any program follows from the premise that a Turing machine is the model for any effective procedure. Any algorithm can be modeled by a Turing machine. Since an "algorithm is just an abstract program,"[141] any program can also be modeled. In fact, even the programming language can be modeled.[142]

3. *Computers Are Really Both*

Both hardware and software can be modeled using the same abstract mathematical model. Both hardware and software can be reified as electronic circuitry and ancillary hardware. The mathematics and the physical reality point to the same conclusion: the legal dichotomy between hardware and software is an artificial distinction.

B. UNIFYING THE ELEMENTS OF COMPUTERS THROUGH A VIRTUAL MACHINE MODEL

Computers, as they appear to the user, are monolithic. The user does not distinguish between hardware features like the data bus and the memory chips.[143] Likewise, a typical user makes no distinction between hardware and software features such as shift registers and microprogramming. Indeed, the user is probably blissfully oblivious to their existence. The virtual machine model captures this perception of the user.

1. *What is a Virtual Machine?*[144]

Virtual machines are recursively constructed.[145] The two components of a virtual machine, machines and languages, are defined in

141. Newell, *supra* note 56, at 1029.

142. "A higher-level formal language is an abstract machine." J. WEIZENBAUM, COMPUTER POWER AND HUMAN REASON 158, 240 (1976) *quoted in* CONTU REPORT, *supra* note 91, at 37 (Hersey, Comm'r, dissenting).

143. Unless, of course, the user intends to manipulate that specific component.

144. *See* Appendix D. As with the glossary, you may wish to pull out the diagrams as a quick reference.

145. Recursion can be illustrated by the Russian dolls that encapsulate smaller dolls. A small solid doll is encased by two pieces that form a larger, but otherwise identical, doll. The resulting doll is likewise encased in a larger doll. The process may, theoretically, be repeated *ad infinitum*. In this model, the hardware is analogous to the original doll.

terms of each other. A machine is an abstract object that does certain things according to the rules of its language. A language is simply the rules followed by the corresponding machine. The recursion comes in when one language is translated into another. If the basic machine is M1 with language L1, and another language L2 is translated into L1 to make M1 perform some task, a new machine, M2, is created. M2 is defined by the rules of L2. Each machine and language constitute a level. M2 may not actually be doing the task in some sense, but, assuming no flaws in translation, it is as if M2 actually were doing the task. This process can be repeated an infinite number of times theoretically. In practice, each higher level enhances some aspects of the machine (such as ease of use) while decreasing other aspects (like speed and memory space). Nevertheless, virtual machines are practical. One common application is a computer that serves several users simultaneously. The operating level of the machine may create several emulations of itself so each user works with an emulation that appears to be the operating level itself, albeit with certain restrictions on memory, speed, and function.

2. *How Does It Apply to Computers?*

A virtual machine can model a computer. In 1951, M.V. Wilkes suggested a three-level machine to simplify hardware. His idea was to use levels to take what is called "machine language"[146] and interpret it, using a built-in interpreter, into the microprogramming language. By the end of the decade, interpreters and compilers were the norm.[147] Now a computer is commonly understood to be a multilevel machine. Hardware by itself is just digital circuitry. It takes several virtual machine levels to reach something that an ordinary use would consider a computer.

C. THE APPLICABILITY OF A VIRTUAL MACHINE MODEL TO THE LEGAL REALM

The virtual machine model captures the continuum of computer innovation from hardware through firmware to software. It also reflects the user's perception of a computer. A model that describes the interrelationships between computer elements *and* describes the consumer's understanding of the product has many lessons for lawyers.

146. This is used in a specific sense, not in the sense of a language for a virtual machine level. This "machine language" is a machine language of a virtual machine level, but the term is not used here in its general sense.

147. A. TANENBAUM, *supra* note 5, at 8.

1. *A Unified Model is Consistent with Modern Law*

Patent law strives for uniform treatment of utility patents.[148] Machine components generally are not treated differently under the patentable subject matter requirement. Processes and machines are routinely patented. This uniformity dissolves when digital computer elements are claimed. As the *Benson* Court explained:

> It is said that the decision precludes a patent for any program servicing a computer. We do not so hold. It is said that we have before us a program for a digital computer but extend our holding to programs for analog computers. We have, however, made clear from the start that we deal with a program only for digital computers. It is said we freeze process patents to old technologies, leaving no room for the revelations of the new, onrushing technology. Such is not our purpose.[149]

Yet the effect was to arbitrarily bifurcate programs for computers from all other programs. In *Flook*, the Court held that the word "process" does not apply to digital computer programs even though the terms are literally interchangeable. The Court justified this term-of-art distinction on the basis that a digital computer program is a "mathematical formula . . . [with] no substantial practical application except in connection with a digital computer."[150] By logical extension, any process that can be mathematically modeled should not be patented.[151] The Court feared granting a patent would preempt a field of mathematics,[152] but, in fact, the patent would only preempt physical realizations of the algorithm in the narrow context of digital computers or, as in the case of *Flook* or *Diehr*, physical manufacturing processes.

2. *The Virtual Machine Model Could be Used in Computer Intellectual Property Law*

The virtual machine model is completely amenable to existing patent law. The model provides machines and translations between machines. This corresponds beautifully with patent subject matter definitions in the Patent Act. Any level of the virtual machine is a separate machine, and could be patented as a machine. Translations between machines are processes, and could be patented as such. The virtual machine also allows a sharp division between the user who perceives the virtual machine and the virtual machine itself. This satisfies Commissioner Hersey's concerns,[153] and makes copyright inappropriate

148. There are isolated exceptions for drugs and plants. 35 U.S.C.A. § 155-56 (West Supp. 1987) (drugs), §§ 161-64 (plants).

149. Gottschalk v. Benson, 409 U.S. 63, 71 (1972).

150. Parker v. Flook, 437 U.S. 584, 589 (1978) (citing Benson).

151. This probably includes all processes. *See* text accompanying notes 135-40.

152. *Benson*, 409 U.S. at 72.

153. *See supra* note 101.

inside a computer.[154]

IV. THE IMPLICATIONS OF THE MODEL FOR COMPUTER INTELLECTUAL PROPERTY LAW

Use of virtual machines to conceptualize the varieties of computer innovation would create order out of chaos. Uniform treatment of computer elements as potentially patentable subject matter would extend to all innovations used in the virtual machine.

A. NO CHANGE IN TREATMENT FOR HARDWARE

Hardware is patentable subject matter under the current regime. The virtual machine model would not change that. Hardware constitutes the base-level machine, so it remains patentable. The changes occur on the subsequent levels.

B. A NEW VIEW OF PROGRAMS

The virtual machine revolutionizes the legal view of programs. Once a program enters a computer, it is part of the machine.[155] As components of a machine, programs should be patentable subject matter.

1. Benson *and* Flook *Are No Longer Good Law*

Diehr implicitly overruled much of the *Benson-Flook* rule. The dissent in *Diehr* echoed *Benson* and *Flook* in its inability to distinguish abstract algorithms from algorithms implemented as programs. Not all algorithms can be reduced to programs.[156] By a narrow majority,[157] the

154. Copyright is inappropriate inside a machine since that would give copyright protection to something that is inherently utilitarian. This conclusion would also seem to follow from Mazer v. Stein, 347 U.S. 201 (1954), in which a lamp manufacturer sought to enforce a copyright in decorative lamp stands as "works of art." The Court concluded that the copyright owner could prevent others from copying its particular lamp stands, but not lamp stands with similar statuettes, and that artistic articles may be protected in form but not in their mechanical or utilitarian aspects. This opinion was endorsed in H.R. REP. NO. 93-1476, 83d Cong., 2d Sess. 54 (1954).

155. This is a bit simplistic in the sense that, in some situations, a program may be more like data than a machine level per se. For instance, most programs are composed on computers and may even someday be composed by computers. In that context, they are data that the computer manipulates. While this may seem like copyrightable expression, a better analogy may be made to an I-beam extruder at a steel mill. The I-beam is created by a machine, perhaps with guidance by a human operator, and is, by itself, merely a sculptural expression. The I-beam takes on utility when used as a structural member, perhaps in the very mill in which it was created. No one would argue that the I-beam should be covered by copyright until it is used. For simplicity, a product intended to be used is considered patentable from its creation.

156. For instance, computer scientists regularly study algorithms that fall within a

Diehr Court allowed an algorithm in the context of an industrial process to be patented as part of the industrial process. While recognizing programs could be patented was a step in the right direction, the Court did not go far enough: the Court should have reinstated all programs as potentially patentable processes.

2. *Programs Should Always be Patentable*

The patent process must carefully differentiate between ideas and inventions. To be an invention, an idea must be reified. The PTO has the authority to demand a working copy.[158] A program or reified algorithm could meet this requirement; an abstract algorithm could not.[159] Algorithms that are thus reducible may be patentable. Professor Newell worries that exceedingly useful algorithms like Dantzig's simplex method, or like Cooley and Tukey's fast Fourier transform will be monopolized.[160] The simple answer is, "Of course, that is the whole point." The intellectual property system encourages useful innovation by rewarding innovators with exclusive rights to their innovation for a limited term of years. In this respect, patent is better than copyright because the patent term is seventeen years, while for copyright the term is at least fifty.[161] The alternative is to offer no protection. Although he acknowledges that a lack of protection is not ideal, Newell avers natural human curiosity will maintain a flow of innovation.[162] While that is undoubtedly true, the same could be said of any area of innovation. Humanity exited, developed technologically, and even thrived before intellectual property law came about. The more appro-

---

class of problems that cannot yet be solved deterministically in polynomial time. *See* J. HOPCROFT & J. ULLMAN, *supra* note 138, at 320-65. These intractable problems could not be reduced to practice and, for that reason, should not be patentable. They are like perpetual motion machines. Nevertheless, they are interesting problems, and scientists will continue to study them.

157. It was a 5-4 decision. The dissenters (Stevens, Brennan, Marshall, and Blackmun) are all still on the Court. Of the majority, only White and Rehnquist remain.

158. 35 U.S.C.A. § 114 (West Supp. 1987)(Models, specimens).

159. For instance, the Hamilton circuit problem cannot currently be solved deterministically in polynomial time. J. HOPCROFT & J. ULLMAN, *supra* note 138, at 332-35. A claim for a method to solve it nondeterministically should fail because it could not be reduced to practice. It would be like patenting a perpetual motion machine. If, on some glorious day, the Hamilton circuit problem is deterministically solved in polynomial time, it will be reducible to a program. This would be one case where novelty, non-obviousness, and utility would not be questioned.

160. Newell, *supra* note 56, at 1028. The simplex method (1948) is a powerful tool used in complex management and production problems. The fast Fourier transform (1965) is at the heart of signal processing technology.

161. 35 U.S.C.A. § 154 (West Supp. 1987), 17 U.S.C.A. § 302 (West Supp. 1987), respectively. *See also* Appendix C.

162. Newell, *supra* note 56, at 1026.

priate question is, "To what extent is intellectual property law good for technological development?" In this question, computer innovations should not be separated from other innovations. Researchers, outside computer science and engineering, labor under the same constraints. If the system fails for computers, then it fails for other types of intellectual property as well.

C. ADVANTAGES OF THE VIRTUAL MACHINE MODEL

The main advantage of the virtual machine model is that it better reflects what a computer actually is. It eliminates the artificial distinctions between hardware and software, and it, instead, reflects the continuum that in fact exists. It provides a bright line, based on the user's perception, demarcating the realm of computer function from the realm of human expression. Components integral to computer function should be patentable, and would be if a virtual machine model were used.

A uniform model creates legal predictability. Business thrives on legal predictability. This certainly could meet the policy goals of federal intellectual property law by encouraging innovation, and by encouraging full disclosure of innovation. Innovation would be encouraged by the certainty of a predictable form of entitlement. Disclosure would be encouraged by the mandatory registration required for patent protection.[163] The disclosure must be "in such full, clear, concise, and exact terms as to enable any person skilled in the art to which it pertains . . . to make or use the [invention or process, and must] set forth the best mode contemplated by the inventor."[164]

This distinction between the computer realm and the user realm is also a better use of federal intellectual property law. Patent is designed to accommodate useful items. This is reflected in the shorter term for patents and in the severe requirements regarding disclosure. With the pace of innovation for computers, few computer innovations are still state-of-the-art after seventeen years, much less seventy-five. Patent law is better suited to the useful nature of computer programs.

D. DISADVANTAGES OF THE MODEL

Patent law is no panacea. Patents are very slow, difficult, and expensive to obtain, and will do little to solve the problem of software piracy. Software piracy is a serious problem for the software industry. Piracy is perpetrated by two distinct groups: consumers and rival com-

163. 35 U.S.C.A. §§ 111-14 (West Supp. 1987).

164. *Id.* at § 112.

panies.[165] Laws against piracy are nearly impossible to enforce against consumers since piracy is easy and widespread, detection is extremely unlikely, and prosecution is expensive.[166] Patent law will be no better at preventing this type of piracy than copyright. Piracy by rival companies will also not be affected much, although the virtual machine model, with its emphasis on function and algorithm, may encourage judges to look to the data structures and algorithms rather than the "look and feel" or expression of the program. Any "look-and-feel" claims that do not qualify as computer components would be relegated to trademark protection.

The general difficulty of obtaining and maintaining patents is insurmountable. It is too hard and too expensive to apply for a patent, and the protection is too long in coming. To some extent, that is good. Trivial enhancements should be weeded out by the novelty and non-obviousness requirements anyway. Trivial innovations should not receive protection from competitors. Copyright does not protect independently developed innovations and, with the burden to prove copying resting with the plaintiff, for trivial innovations the effect of patent and copyright is substantially the same. For more significant innovations, however, the problem remains.

Some relief is possible for innovators who simply wish to retain their own right to use the innovation without excluding others.[167] The Patent Act provides for statutory registration of inventions.[168] The requirements for statutory registration are much easier than those for patent. The applicant must simply meet the disclosure requirements discussed earlier, and must waive the right to receive a patent on the invention.[169] Another possibility is to publish or otherwise widely disseminate a description of the invention before anyone else invents it.[170] An inventor desiring an exclusive right, however, is still at the mercy of the patent system.

On the other hand, the high cost of obtaining a patent could be considered a market force that separates those innovations deserving fed-

165. Frysinger, Three Strikes You're Out? Software Companies and Consumers Lose in the Computer Ball Game (Apr. 1, 1988) (unpublished manuscript).

166. *Id.* at 7-8. *See generally* Note, *Software Piracy and the Personal Computer: Is the 1980 Software Copyright Act Effective?*, 4 COMPUTER/L.J. 171 (1983). Indeed, intense prosecution of piracy lawsuits against consumers might even injure the software developer's reputation since piracy is so widespread. The result is similar to a policy of denouncing drug use, but not doing much to catch the users.

167. Many manufacturers may think that they can retain their market share without a patent because they have better marketing abilities, etc. Nevertheless, they will want to avoid being shut out by a rival's patenting of an innovation they currently use.

168. 35 U.S.C.A. § 157 (West Supp. 1987) (Statutory invention registration).

169. *Id.* The disclosure requirements are in § 112.

170. *Cf. id.* at § 102(a).

eral intellectual property encouragement from those that are insignificant innovations. If the innovation is meritorious, some person or corporation should be willing to finance the patent application. The PTO is slow and fastidious. This is a problem for all patent applicants and should be remedied regardless of whether the patent is for computer innovations or for better mousetraps.

V. CONCLUSION AND RECOMMENDATIONS

In our culture, computers have considerable mystique. We refer to them as thinking machines and expect them to be at least as capable as we humans, if not more so. The *Benson* Court describes a computer as "solving a problem by doing arithmetic as a person would do it by head and hand."[171] There is every indication the Court takes this literally. If so, it can only impede our ability to work with computers. Computers are just machines[172] and should be treated as such by our federal intellectual property system. The virtual machine could be a useful tool for demystifying computers. It cannot, however, magically improve the patent system.

The Supreme Court has repeatedly sought a legislative solution, recognizing Congress as the most competent branch for settling the matter.[173] Unfortunately, the only congressional action was the formation of a commission to study computer programs separately in the context of copyright.[174] This prejudicial assignment compounded the Court's error. Only Commissioner Hersey, who relied heavily on the writings of a computer scientist, recognized the magnitude of that error.

Since then, criticizing the *Benson-Flook* case line and the CONTU Report has become a hot topic in intellectual property law.[175] Virtually all of the authors suggest some sort of *sui generis* solution, often modeled after the Semiconductor Chip Protection Act.[176] The virtual machine model shows *sui generis* solutions are both unnecessary and unwise. Special treatment of each element of a computer would hopelessly complicate the law without any chance of covering every possible

171. Gottschalk v. Benson, 409 U.S. 63, 65 (1972).

172. Steve Jobs introduced Apple Corp.'s Macintosh as an "information appliance."

173. *Benson*, 409 U.S. at 73; Parker v. Flook, 437 U.S. 584, 594 (1978).

174. That commission was CONTU. *See supra* note 91.

175. *E.g.*, Samuelson, *supra* note 12; Note, *supra* note 166; Chisum, *The Patentability of Algorithms*, 47 U. PITT. L. REV. 959 (1986).

176. E.g., Harris, *supra* note 105 (recommending protection for microcode like the protection conferred in the Semiconductor Chip Protection Act); *contra* Raskind, *The Uncertain Case for Special Legislation Protecting Computer Software*, 47 U. PITT. L. REV. 1131 (1986) (all that is needed is congressional hearings on whether the Copyright Act needs a little fine tuning).

case. While patent law is no panacea, there is no problem with patents that is unique to computer components.

If Congress chooses to act again in this area, it should not direct its attention to exotic, narrow topics like *sui generis* protection for microcode; instead, it should focus on the broader and more mundane topic of patent reform. This would benefit all potential patentees and would have the effect of truly promoting the progress of the useful arts. If a *sui generis* solution is necessary anywhere, it may be in the general category of process patents. Processes, software and otherwise, all tend to share problems like ease of appropriation and difficulty of effective protection.[177] If Congress wants to help out, it should investigate ways of making patents easier, faster, and cheaper to attain.

Two possible solutions to Professor Newell's concern about intellectual property strangleholds on research might be compulsory licensing or fair-use rules for non-commercial endeavors.[178] Thorough congressional consideration may reveal a way to let academics and other non-commercial users have their cake while the patentees exploit it. This concern may also be something of a chimera. Researchers in other fields do not appear to be stymied by intellectual property law, except when secrecy bogs down attempts by other researchers to verify claimed results.[179]

What ever else Congress may do, it should pursue simpler models. Computer programmers have an acronym: KISS. It stands for Keep It Simple, Stupid. One of the greatest temptations in computer programming is to add a lot of frills while ignoring the substance of the program. This is also a problem with computer intellectual property law. The courts and the commentators have unnecessarily multiplied the types of innovations and the types of protections. We would do better if we simply kept it simple.

---

177. *Cf.* Corning Glass Works v. Sumitomo Elec. U.S.A., 671 F. Supp. 1369 (S.D.N.Y. 1987) (Sumitomo and its American subsidiaries used Corning's patented processes for making several types of optical fibers using a specific method with restrictions on the types of, and a critical limit on the maximum amounts of, dopants).

178. *See* Chisum, *supra* note 166.

179. *See, e.g.*, Lemonick, *Fusion Illusion?*, TIME, May 8, 1989, at 74. In the case of computer science, patents would probably be an improvement because copyright does not protect the algorithm itself so really sophisticated algorithms tend to be hidden behind trade secrets. With patents, useful applications on an algorithm could be proteced, but the algorithm itself would have to be clearly disclosed.

APPENDIX A: GLOSSARY

Terms appear as they were defined in the text or in footnotes.

ANALOG COMPUTER — A mechanical, non-electronic, computer. Babbage's analytic engine was an analog computer.

ALGORITHM — a) A step-by-step procedure for solving a problem or accomplishing some end; b) an unambiguous specification of a conditional sequence of steps or operations for solving a class of problems; c) a computer program.

HARDWARE — Tangible objects including integrated circuits, printed circuit boards, cables, power supplies, memories, card readers, line printers, terminals, etc.

FIRMWARE — Software embedded in electronic devices (hardware). See MICROPROGRAMMING.

MASK — A stencil used to manufacture chips by means of etching and depositing on semiconductor wafers.

MICROCODE — See MICROPROGRAMMING.

MICROPROGRAMMING — Software embedded in firmware to control the hardware aspects of the device. For example, a microprocessor chip like the Motorola MC68000 or the Zilog Z80 is a piece of metal, plastic, silicon and trace elements. Those materials constitute the hardware. In addition, to avoid further complexity in the hardware, the designers built into the hardware certain very low-level programs. These programs are the microprograms. They are also called microcode.

OBJECT CODE — Object code is source code that has been converted by a program called a compiler into machine-readable instructions. Object code needs to be "linked" by a program called a linker to other bits of object code and to certain parts of the computer's memory before it can be used.

SOFTWARE — Algorithms and programs, their computer representations.

SOURCE CODE — Source code is a program written by a programmer in a high-level language, i.e. a mnemonic language, like Fortran, BASIC, Pascal, or C.

TURING MACHINE — A Turing machine, is purely a mathematical model. It posits a tape, infinitely long in one direction, divided into cells containing symbols from a finite set of symbols. A control device with a reading and writing head scans the tape. The control device has a finite number of states it can be in and a set of rules for moving. Turing machines can model any effective procedure.

VIRTUAL MACHINE — A model for computers consisting of "machines"

connected by "languages." A machine is a level which appears as a computer to the user at that level. A language is the means by which adjacent levels communicate.

APPENDIX B: PATENT EXAMINING PROCEDURES

Excerpted from the U.S. PATENT AND TRADEMARK OFFICE, MANUAL OF PATENT EXAMINING PROCEDURES § 2106 (5th ed. 6th rev. Oct. 1987). This is the P.T.O's description of the rules outlined in *Diehr*, hence this is the current examining standard. Rule five is a carryover from *Flook*. It is unnecessary under a virtual machine model.

1. The "claims must be considered as a whole. It is inappropriate to dissect claims into old and new elements and then to ignore the presence of the old elements in the analysis." . . . "The 'novelty' of any element or steps in a process, or even of the process itself, is of *no relevance* in determining whether the subject matter of a claim falls within the 101 categories of possibly patentable subject matter" (emphasis added).

2. "When a claim containing a mathematical formula implements or applies that formula in a structure or process which, when considered as a whole, is performing a function which patent laws were designed to protect (e.g., transforming or reducing an article to a different state or thing), then the claim satisfies the requirements of § 101."

3. "When a claim recites a mathematical formula (or scientific principle or phenomenon of nature), an inquiry must be made into whether the claim is seeking patent protection for that formula in the abstract." (If the claim does seek protection for such a mathematical formula, it would be non-statutory under 35 U.S.C. 101).

4. "A mathematical formula as such is not accorded the protection of our patent laws . . . and this principle cannot be circumvented by attempting to limit use of the formula to a particular technological environment." . . . "Similarly, insignificant post solution activity will not transform an unpatentable principle into a patentable process."

5. When a claim as in *Parker v. Flook*, 198 USPQ 193 (1978), is drawn "to a method for computing an 'alarm limit' (which) is simply a number," the claim is non-statutory under 35 U.S.C. 101 because *Flook* "sought to protect a formula for computing this number."

6. "It is now commonplace that an *application* of a law of nature or mathematical formula to a known structure or process may well be deserving of patent protection." [Citations omitted.]

APPENDIX C: CONTU REPORT, TABLE 1

This Appendix reproduces Table 1 from CONTU Report at 19.

CHARACTERISTICS OF PROTECTIVE MECHANISMS

CONSIDERATIONS	COPYRIGHT	PATENT	TRADE SECRECY
General			
National Uniformity	Yes	Yes	No
Protection effective upon	Creation of work	Successful prosecution of application	Entrance into contractual relationship
Cost of obtaining protection	Nil	Moderate	Moderate
Terms of protection	Life plus 50 years or 75 years	17 years	Possibility of both perpetual protection and termination at any time
Cost of maintaining protection[1]	Nil	Nil	Significant
Cost of enforcing rights against violators[2]	Moderate	Moderate	Higher
Availability of (a) statutory damages and (b) attorney's fees from infringers	(a) Yes (b) Yes	(a) No (b) Yes	(a) No (b) No
Protection lost by	Gross neglect	Unsuccessful litigation	Disclosure
Software, including effects of Commission proposals			
Consistency with other copyright areas	Yes	No	No
Availability of protective mechanisms for some programs[3]	Yes	Unclear	Yes
Universal availability of protective mechanisms for all programs[4]	Yes	No	No
"Process" protectible	No	Yes	Yes
Suited to mass distribution	Yes	Yes	No

[1] Once copyright or patent is secured, it costs little or nothing to keep it in force; on the other hand, expensive security measures must be taken to avoid losing a trade secret. At least part of the cost of this security is passed on to the user.

APPENDIX C: CONTU REPORT, TABLE 1 (cont.)

[2] Copyright and patent infringers in some instances may be persuaded to comply without the institution of a lawsuit. If litigation is necessary, it may be expensive, but in copyright and patent cases, attorney's fees may be awarded to successful plaintiffs. At trial, the proprietor bears the burden of proving that the trade secret is valid; in patent cases, there is a presumption of validity; and in copyright actions, a registered certificate is prima facie evidence of a copyright's validity. The proof of the validity of a trade secret may be expensive and difficult, as it almost necessarily involves retention of expert witnesses. Although a witness may be needed in copyright and patent suits, in those cases there will have been at least some compliance with federal law regarding public notice of claimed rights before the suit is initiated. A suit to enforce a trade secret, even though successful, may destroy the secret if it is offered into evidence and becomes part of the public record of the trial.

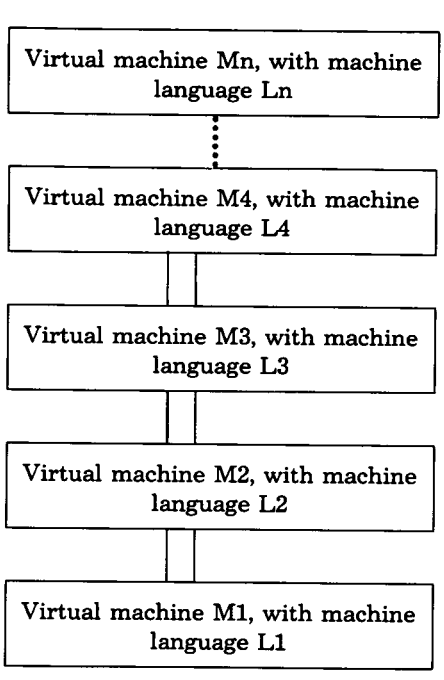
[3] As of the present, serious doubt exists whether programs are proper subjects for patent protection. (See this chapter under Copyright and other Methods Compared). [Note: This chart was made before *Diehr*.]

[4] Even if programs are patentable, only those that are truly novel and non-obvious will be protected. [I believe this is the preferred situation.] Trade secrecy is, of course, unavailable when the contents of a program have been disclosed.

APPENDIX D: VIRTUAL MACHINES ILLUSTRATED

These diagrams are based on figures 1-1 and 1-2 in Andrew Tanenbaum's *Structured Computer Organization* 3 & 5 (1984).

A MULTILEVEL MACHINE



Programs in Ln are either interpreted by an interpreter running on a lower machine, or are translated to the machine language of the lower machine.

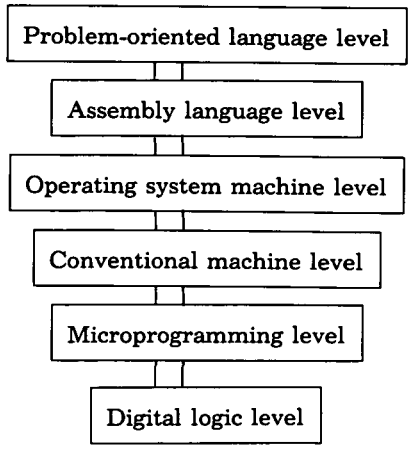
Programs in L4 are either interpreted by an interpreter running on M3, M2, or M1, or are translated into L3, L2, or L1.

Programs in L3 are either interpreted by interpreters running on M2 or M1, or are translated to L2 or L1.

Programs in L2 are either interpreted by an interpreter running on M1, or are translated to L1.

Programs in L1 can be directly executed by the electronic circuits.

THE SIX LEVELS OF A TYPICAL MODERN ELECTRONIC COMPUTER



Translation (compiler).

Translation (assembler).

Partial interpretation (operating system).

Interpretation (microprogram).

Microprograms executed directly by the hardware.