Appendix A
Legal Protection for Computer Software

Computer **software** can be protected under copyright patent or trade secret law, or under some combination of these. This appendix briefly reviews these forms of protection, with emphasis on applications to computer software.

A related, sui generis, form of protection for semiconductorchip mask designs is provided via the Semiconductor Chip Protection Act of 1984.[1] The Act protects the designs of the mask works used to lay out integrated circuit designs in semiconductor chips.[2]

Copyright

The current copyright law is enacted in the Copyright Act of 1976, as amended (Title 17 U. S.C., ch. 1-8,90 Stat. 2541). A 1980 amendment made explicit provisions for computer programs as (literary) works of authorship (Public Law %-517, 94 Stat. 3-15, 3028). Copyright protects "original works of authorship" from ***unauthorized uses*** including reproduction (copying), making derivative works (adaptation), public distribution, public performance, and display.[3] Generally, the term of copyright for new works is the life of the author plus 50 years, or 75 years for works made for hire (e.g., by art employee of a firrn).[4]

Copyright has been the form of software protection favored by most nations (see app. B). Obtaining a copyright is easy, inexpensive, and quick compared to the requirements for obtaining a patent (see next section on patents). Since copyright is administered under Federal law, unlike trade secret protection, it is uniform in all the States. The duration of copyright protection is very long, compared to the expected economic or technical lifetimes of computer programs.

The doctrine of fair use is one of several statutory limitations on copyright holders' exclusive rights. Under this doctrine, certain unauthorized uses, such as copying for the purposes of teaching, scholarship, or research, may be considered "fair use:' not copyright infringements. Whether an instance of copying is a fair use instead of an infringement is determined by the courts, taking four statutory criteria into account: 1) the purpose and character of the use, 2) the nature of the copyrighted work, 3) the amount and substantiality of the portion used in relation to the work as a whole, and 4) the effect of the use on the potential value of or market for the work.[5]

Another statutory limitation on the rights of software copyright holders is given by section 117 of the copyright law, added in the 1980 amendment:

> . . . it is not an infringement for the owner of a copy of a computer program to make or authorize the making of another copy or adaptation of that computer program provided:
>
> 1. that such new copy or adaptation is created as an essential step in the utilization of the computer program in conjunction with a machine and that it is used in no other reamer, or
> 2. that such new copy or adaptation is for archival Purposes Only and that all archival copies are destroyed in the event that continued possession of the computer program should cease to be rightful.

This limitation clarifies the right of a user who legitimately owns a software product to make "backup" copies of the software to protect against damage or loss, to load the software onto the hard disk of a computer for easier or more efficient use, and to make adaptations if necessary to make the program usable on a computer (e.g., compiling it, inserting default formats or directory paths,

[1]Public Law 98-62(1,98 Stat. 3347,3356.

[2]See Cary H. Sherman, Hamish R. Sandison, and Marc D. Guren, *Computer Software Protection Law* (Washington, DC: The Bureau of National Affairs, Inc., 1989), part 300. Protection for an original mask work extends for 10 years from the time it is registered; the duration takes into account the relatively short useful economic life of a particular chip. The Act was developed in a period when chip design and mask-work production was a very labor-intensive and time-consuming step in chip manufacture, so that protection from copying of the mask work conferred a significant competitive advantage. The Act does not provide protection for the underlying idea, or against independent creation, reverse engineering, or instances where the design was not copied. Patent protection also applies to chips, but patents require public disclosure of the invention. Also, the layout of a chip will rarely satisfy the levels of novelty and nonobviousness required by patent law, which protects invention, not effort.

[3]An "original work" is one that does not have the same expression as a preexisting work; an identical, but independently created, work is not a copyright infringement. (The "originality necessary to support a copyright merely calls for independent creation, not novelty." Melvin B. Nimmer, N*immer on* Copyright (New York, NY: Matthew Bender, 1982), vol. 1, sec. 201(A), cited in U.S. Congress,Office of Technology Assessment, *Intellectual Property Rights in an Age of Electronics and Information* (Melbourne, FL: Kreiger Publishing Co., April 1986), OTA-CIT-302, ch. 3, footnote 10.) chapter 3 of the 1986 report discusses intellectual property concepts.

A "derivative work" is a work based on one or more preexisting works, such as a translation, abridgement, condensation, etc. (Copyright Act, sec. 101).

[4]See Copyright Act, sec. 302. For further discussion of U.S. copyright law, see U.S. Congress, Office of Technology Assessment, *Copyright and Home Copying: Technology Challenges the Law*, OTA-CIT-422 (Washington, DC: U.S. Government Printing Office, October 1989), ch. 3.

[5]Copyright Act of 1976, ch. 1, sec. 107. Computer software can present some particular problems in assessing what is "fair use." For example, a competitor who de-compiles a copyrighted object-code program in order to study the unprotected ideas it embodies will necessarily make a "copy" of the entire program (see app. A, footnote 7).

etc.). It does not permit making and distributing multiple copies for school or office use.

Copyright does not confer rights over ideas-only the *expression* of an idea is protected, not the underlying idea itself.[6] A copyright holder (e.g., a software developer) might consider this to be a disadvantage, because his copyright will not preclude a competitor from creating a new work embodying the same idea, so long as the competitor does not incorporate copyrighted expression from the first program into the second program. For software, copyright may also allow "clean room" reverse-engineering practices.[7] In this type of reverse engineering, one team of software developers studies the code of a copyrighted program to extract the underlying functionality (ideas). A second team (who has never had access to the copyrighted code) then creates a new program, based on the first team's functional specifications. For some computer software, writing the code may be relatively trivial, so that the true innovation and market advantage lie in the program's logical structure or in its interfaces. The extent to which these are protectable expression, as opposed to uncopyrightable ideas, is the focus of the latest round of court cases.

Disputes Over Copyrightability

There has been considerable disagreement over what features of a computer program are (or should be) copyrightable. The distinction between idea and expression can be very tricky to make, even for some traditional literary works like books and plays. For software, which is intrinsically functional, idea and expression are closely interwoven, even in theory. In practice, it is extremely difficult to separate which elements of a program are the expression and which are the underlying idea.[8] There is substantial disagreement among legal scholars and among software developers and computer scientists as to whether copyright should protect only against literal or near-literal copying (e.g., mechanical translations, paraphrasing, and disguised copying), or should also protect a program's structure, sequence, and organization and user interfaces (including "look and feel") as well.[9] For example, some in the computer and legal professions believe that a program's "look and feel" should not be protected by copyright instead, these individuals think that protection for "look and feel" is better suited to a patent framework.[10] Others, however, are critical of patent protection for computer processes in programs.[11]

Many in the computer and legal professions believe that "traditional rules of copyright law adapt very comfortably" to new forms of expression like computer latest programs.[12] These individuals believe that there is considerable room for expression in even detailed aspects of a program like its design, logic, structure, and flow,[13] and that the courts can be and generally have been successful in adapting traditional copyright principles to software-infringement cases, even those involving idea-v.-expression or structure, sequence, and organization questions.[14] Therefore, they think that copyright is viable—and vital-as a vehicle for protecting computer

[6] "In no case does copyright protection for an original work of authorship 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 in such work." (Copyright Act of 1976, ch. 1, sec. la(b).)

[7] One court decision has found copying for the purpose of reverse engineering to be sanctioned by section 117 of the Copyright Act. (*Vaults* v. *Quade*, 655 F.Supp. 750, E.D. LA. (1987), affirmed, 847 F.2d 255 (1988), cited by Brian Kahin, personal communication, *Dec.* 1, 1989).

[8] For example, the decision in *Whelan Assoc. Inc.* v. *Jaslow Dental Laboratories, Inc.* (797 F.2nd 1222 (3rd Cir. 1986), cert. denied, 107 S. Ct. 877, 1987) held that the underlying purpose of a program is its "idea," and everything else is expression, given that more than one way to achieve the purpose is possible. Under this interpretation, virtually any elements of the program's structure, sequence, or organization would be considered copyrightable. By contrast, the decision in *Plains* Cotton *Coop*. Assoc. v. *Goodpasture Computer Service, Inc.* (807 F.2nd 1256 (5th Cir.), cert. denied, 108 S, Ct. 80, 1987) held that only line-by-line program design or literal code were protectable. (David C. Godbey, "Comment: Legal Documents As A Metaphor for Computer Programs in Copyright Analysis—A Critique of *Whelan and Plains Cotton*," *The Computer Lawyer*, vol. 6, No. 8, August 1989, pp. 1-10.)

[9] Recent court decisions have varied in determining the extent to which program structure, sequence, and organization should be protected by copyright.

The term "look and feel" originated in an article that focused attention on software user interfaces (Jack Russo and Douglas K. Derwin, "Copyright in the 'Look and Feel' of Computer Software," *The Computer Lawyer*, vol. 2, No. 2, February 1985). There is no statutory or case-law definition, although a kindred phrase, "total concept and feel, ''has been adopted by appellate courts, (Pamela Samuelson, "Why the Look and Feel of Software User Interfaces Should Not Be Protected By Copyright Law," *Communications of the ACM*, vol. 32, No. 5, May 1989, pp. 563-572.)

[10] The argument is that "look and feel" is more idea and concept than expression. In that case, however, an innovation that did not represent a novel and nonobvious advance over prior work would not be patentable (see section on patents). Thus, most user-interface improvements would not be protected under either patent or copyright if "look and feel" is not accepted by tlcourts. See: "Computer Scientists Protest SoftwareLitigation," *International Computer Law Adviser*, June 1989, p. 22; and Pamela SamuelSon, ibid.

[11] For example, see Brian Kahin, "The Impact of Software Patents," *Educom Review*, winter 1989, pp. 28-31.

[12] For a discussion of this position and a rebuttal of opposing views, see Anthony L. Clapes, Patrick Lynch and Mark R, Steinberg, "Silicon Epics and Binary Bards," *UCLA Law Review*, vol. 34, June-August 1987, pp. 1493-1594 (seep. 1501).

[13] See Clapes et al., ibid., pp. 1549-1558.

[14] See Clapes et al., ibid., especially pp. 1546-1554 and 1575-1584. See also Morton David Goldberg and John F. Burleigh, "Copyright Protection for Computer Programs: Is the Sky Falling?" *AIPLA Quarterly Journal*, vol. 17, No. 3, 1989. pp. 2%-297. Goldberg and Burleigh argue that even if not all court cases have been correct or clearly articulated, the same is true ofpatent cases forsoftware-related inventions and would be true for any *sui generis* forms of protection (ibid., p. 2%).

software, and that arguments for hybrid or sui generis protections are based on faulty premises.[15]

Before the current copyright law (and 1980 amendment), there was considerable disagreement as to whether programs could be copyrighted as writings and, if so, what forms of computer software were copyrightable-e.g., whether only the higher-level-language (or "source") code could be copyrighted, as opposed to code in assembly language or machine language (the "object" code). Some arguments-which may have distracted attention from more fundamental issues—were based on the presumed inability of humans to read lower level languages or binary object code; according to this rationale, only higher level languages expressed "writings" (for human readers) eligible for copyright protection.[16] These arguments were misguided because human programmers can and do read programs, albeit with more difficulty, in assembly language and machine language.[17]

The 1980 amendment with reference to programs as statements used "directly or indirectly" in a computer (sec. 101) and "adaptations" for purpose of use in a computer (sec. 117), as well as the explicit 1976 provisions for works that can be perceived/reproduced/communicated "either directly or with the aid of a machine or device" (sec. 101), resolved much of this confusion. Court cases have held that computer programs source code, object code, microcode,[18] flow charts, and audiovisual screen displays-are protected.[19]

Patent

A patent protects an invention, ***including application of the underlying idea,*** from copying and from independent creation for a period of 17 years. It protects against literal infringement (making, using, or selling the claimed invention) and also against infringement by equivalent inventions, whether or not the infringing inventor had prior knowledge of the patented invention. The statutory subject matter of a patent is limited to a process, machine, article of manufacture, or composition of matter that is novel, nonobvious, and useful, or to new and useful improvements to these classes of patentable subject matter.

The requirements for a ***patentable*** invention are relatively stringent; patents don't reward hard work per se. The patent requirements for novelty and nonobviousness are a finer screen than the "originality" criterion of copyright. (All "original" software is eligible for copyright, as with any other statutory work of authorship, and copyright inheres in a work as soon as it is created.) Although patents are being granted for software-related inventions,[20] only a small fraction of software is likely to contain a computer process meeting the tests of novelty and nonobviousness.[21]

---

[15] See Clapes et al., op. cit., footnote 12, especially pp. 1501-1505, 1548-1561, and 1583-84. See also Goldberg and Burleigh, ibid., pp. 317-322.

[16] The rule that a work must be readable by a human audience had its *origins* in *White-Smith Music Publishing Co.* v. *Apollo Music Co.*, 209 US. 1 (1908) which ruled that player piano rolls could not be copyrighted. For a discussion of the readability requirement see "Copyright Protection of Computer Program Object Code, ''Harvard Law *Review,* vol. 96, May 1983, pp. 1723-1744., Christopher M. Mislow, "Computer Microcode: Testing the Limits of Software Copyrightability," Boston *University Law Review, vol.* 65, July 1985, pp. 733-805., and the dissent of Commissioner Hersey in the National Commission on New Technological Uses of Copyrighted Works (CONTU), Final Report, July 31, 1978.

[17] A source program is the program as written by *the* programmer. Writing in lower-level languages like assembly language can be tedious, so programmers usually use a higher-level language like Fortran. For example, a Fortran instruction to add an input "V" to a variable "SPEED" would be ***SPEED = SPEED+ V.*** A Fortran program must be compiled before it is executed by the computer; the compiler translates each Fortran instruction into many binary machine-language instructions.

Similarly, a program written in assembly language must be assembled before it is executed. An assembly-language program generally consists of symbolic statements, each one of which corresponds to one basic operation of the computer. For example, to add "V" to "SPEED" would require statements like LD ***R0,SPEED*** (load SPEED into Register O), LD ***R1,V*** (load V into Register 1), ***AD R0,R1*** (add contents of Register 1 to Register O). Assembly-language programs can't be directly understood by the computer, so an assembler has to translate them into machine language.

In the 1950s-1960s, computer programs were usually entered in the computer in the form of punched cards. As this "source" deck was keypunched, the 80 characters of code on each card were printed at the top for verification and debugging purposes. When the program was compiled, the resulting "object" deck contained only punched holes. This may have contributed to the assumption that object-code programs could not be read by humans.

[18] Microcode governs the operation of the computer within one cycle of the computer's internal clock; itis part of the computers operating system. Copyrightability of microcode was upheld in *NEC Corp.* v. *Intel Corp.* (645 F. Supp. 590 (N.D. Cal. 1986) vacated, 835 F.2d 1546 (9th Cir. 1988).

[19] Sherman et al., op. cit., footnote 2, sees. 203.5(c)-203.7(c).

[20] In the United States, certain types of computer-implemented processes and algorithms can be patented. The Supreme Court has not ruled on whether computer programs per se are patentable subject matter, but has ruled that computer-implemented algorithms that are deemed "mathematical algorithms" per se are not statutory subject matter. Federal courts have thus held that a computer processor algorithm is statutory subject matter unless it falls within a judicially determined exception like the one for''mathematicat algorithms" per se. (See U.S. Patent and Trademark Office, "Patentable Subject Matter: Mathematical Algorithms and Computer Programs," 1106 O.G. 4, Sept. 5, 1989).

In this paper, OTA sometimes uses phrases like "patents for software-related inventions,"" software-related patertts," or "patenting algorithms" to refer generally to patent protection for computer-implemented processes and algorithms. The U.S. Patent and Trademark Office (pTO) considers terms like "software patents" to be a misnomer because they may be interpreted to mean that a computer program per se (i.e., the sequence of coded instructions itself) is patentable, as opposed to the underlying computer process it carries out. (M. Keplinger, G. Goldberg, and L. Skillington, PTO, comments on draft paper, Dec. 18, 1989, pp. 1-2.)

[21] One estimate from the World Intellectual Property Organization places the fractional 1 percent. (Cited in Ingrid M. Arckens, "Obtaining International Copyright Protection for Software: National Laws and International Copyright Conventions," *Federal Communi cations Law Journal,* vol. 38, August 1986,, p..285.)

An advantage of patent protection for the discoverer of a software-related invention is that the patent will protect all the claims for the invention, taken as a whole. (Many of these processes would likely not be protectable under copyright because they would be considered part of the unprotected "idea.") A single computer program may consist of a number of patentable processes and algorithms. At the same time, the claimed invention might be executed by a number of copyrighted programs. Depending on how carefully claims are constructed, the computational logic and processes-even the algorithm itself—Carl be protected.

The United States Patent and Trademark office (PTO) issues patents on inventions that are determined to meet statutory requirements (see above), the first of which is statutory subject matter. Because of the judicially created exception for "mathematical" algorithms (see footnote 20), computer processes that are solely "mathematical" algorithms- "mathematical" ***algorithms per se—are*** not considered to be statutory subject matter. However, ***software-related inventions claiming a new or improved process*** (which can include an algorithm, perhaps even a "mathematical" one) can be statutory subject matter if the patent claims excluding the "mathematical" algorithm are otherwise statutory. Therefore, they can be patented if the other requirements of novelty and nonobviousness are m e t[z]

From the viewpoint of the software industry and society as a whole, some unattractive elements of patents for software-related inventions are procedural, and have to do with the "prior art" and patent searches. The prior art is the body of publicly known technical information against which the patentability of an invention is evaluated. Even if a discovery is "novel" compared to the prior art it must also be "nonobvious." This means that 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,' then the invention is not patentable.[23]

In principle, prior art consists of inventions previously known, sold, or used, including those described in other patents and published articles (see 35 U.S.C. 102). In practice, prior art is most often previously issued patents. Because the bulk of software continues to be protected by copyright and/or trade secret because much of the history of software development is not in the published literature and because relatively few patents for software-related inventions were granted prior to the 1980~ the available and locatable prior art is less complete and relatively more difficult to compile or search than the prior art for other technical fields. PTO classifies patents for software-related inventions according to the field of the process chimed, making it difficult to find or track patents for computer-program processes. Also, nonpatent prior art for software-related inventions is often in nonwritten form, existing only as software products. Thus, there is more risk that invalid patents may issue for widely known or "obvious" computer-program processes.

Patents under examination are not disclosed, so a competitor may put considerable effort into developing a program that unknowingly duplicates computer processes for which one or more patents are pending. The problem of timing and product life cycles is not unique to the computer and software industries, but it is especially troublesome in industries as fast-paced as these. Finally, the process of getting a patent is expensive and lengthy, compared to copyright or trade secret protection. Although turnaround time in PTO is decreasing, a patent still may take years to issue in an industry where products have short economic lifetimes. Enforcement can be difficult and time-consuming, and litigation for infringement runs the risk of finding one's patent invalid.

Problems With Terms and Models

Use of the term "algorithm"[25] has been subject to controversy, largely because computer scientists, lawyers, and the courts have used different definitions of computer-related terms and different models of how "programming" is done. Often, the legal definitions or

[22]The Supreme Court has not ruled as to whether computer programs per se constitute patentable subject matter. Currently, PTO patent examiners carry out a two-part test for mathematical-algorithm statutory subject matter; the test is intended to be consistent witb legislative history and case-law. For examination purposes, "mathematical algorithms" are considered to refer to "methods of calculation, mathematical formulas, and mathematical procedures generally," and no distinction is made between man-made mathematical algorithms and mathematical algorithms representing discoveries of scientific principles and laws of nature (which have never been statutory subject matter). For a process claim involving a mathematical algorithm to be patentable, the claim excluding the algorithm is required to be statutory subject matter-i.e., the claim must be for a process, machine, etc. Trivial post-solution activity like displaying a number is not sufficient. (Patentable Subject Matter: Mathematical Algorithms and Computer ProgramsJ' 1106 0.G.4, Sept. 5, 1989; also contained in *Patent Protection for Computer Software: The New Safeguard,* Michael S. Keplinger and Ronald S. Laurie, (eds.) (Englewood Cliffs, NJ: Prentice Hall Law and Business, 1989), pp. 9-42.)

[23]35 U.S.C. Section 103. See Irving Kayton, *Kayton on Patents* (Washington, DC: Patent Resources Institute, Inc., 1983), ch. 5.

[24]PTO categorizes patents by some 350 classes, each with some 350 subclasses; classification is done according to structural elements. Many software-related patents are classified in classes 364,235, and 340, but not all patents in these classes are software-related. PTO places patents drawn solely to computer processes that are not classifiable in other areas of technology in Class 340, Subclass 300. (M. Keplinger, G. Goldberg, and L. Skillington, PTO, comments on draft paper, Dec. 18,1989, p. 4)

[25]A common definition of the term *algorithm is:* "a set of ruks which specify a sequence of actions to be taken to solve a problem [or carry out a process]. Each rule is precisely and unambiguously defined so that in principle it can be carried out by machine." (*Chambers* Science and Technology Dictionary, Peter M. B. Walker (cd) (New York, NY: W & R Chambers, Ltd., 1988), p. 23.)

interpretations have been inexact or at odds with common use of the terms by mathematicians, computer scientists and programmers. Thus, legal battles have included arguments over distinctions between "mathematical" and "nonmathematical" algorithms, mathematical algorithms and "numerical" equations, equations and "laws of nature" or "basic truths," algorithms and "mental steps," etc.[26]

While algorithms may be numerical or non-numerical, algorithms are all "mathematical" constructs. Therefore, making distinctions between mathematical and non-mathematical algorithms, or even between algorithms and computer programs, is problematic for the long term.[27] Moreover, while a particular algorithm may describe a new or improved method of carrying out an operation (like a Fourier transform), or even the most computationally efficient, it may not describe the only method.

Trade Secret

Trade secret law protects the owners of certain information against its misappropriation. Others, who have not obtained the information by improper means, are free to use the information and associated ideas. Unlike copyright or patent, there is no limitation on its duration. Trade secret has been the traditional favorite form of protection for mainframe and minicomputer software. From the viewpoint of a software developer, the advantages to trade secret are that it protects a program's underlying ideas, logic, and structure, not just expression (as in copyright). It avoids formalities of registration or application and lengthy waits for protection. Enforcement is relatively clear-cut and injunctions or compensatory relief is available for those who can prove misappropriation of trade secrets.

On the other hand, trade secret protection doesn't protect against independent creation, reverse engineering, or accidental disclosure of the secret. Also, it can be costly or impossible to maintain secrecy, and the lack of uniformity in State and national laws can be frustrating. In the United States, trade secrets are protected by individual State laws, although there is a Uniform Trade Secrets Act enacted in many States with minor variations. (Although most developed nations have some form of protection for confidential business information, most foreign nations outside of Western Europe do not have trade secret laws per se.) However, much of what trade secret law does can be accomplished by contract and by enforcing licensing terms against disclosure.

When software is protected by trade secret it maintains that status so long as it is not publicly disclosed. For society, this can lead to a lack of knowledge about the state-of-the-art.[x] In turn, this can adversely affect prior art for patent examinations and lead to "reinventing the wheel" rather than building on (or around) prior advances.

Maintaining Software as a Trade Secret

For software (or anything) to be protected as a trade secret, it must not be generally known to a competitor, there must bean effort to maintain its "secrecy," and **those** to whom the secret is disclosed must have a duty not to mistreat the information. However, if they do and a third party gets hold of it, then in many jurisdictions, the third party has no duty to respect the trade secret.

Trade secret protection became popular long before the wide proliferation of personal computers (PCs). Markets (for mainframe and minicomputer software) were smaller then, and much software was custom-developed for particular clients or small market niches.

Some parts of the software market still work like this. In these types of markets, trade secret software has relatively limited exposure, usually to users with contractual obligations to the developer. Often, software is delivered to the client in a lower-level language like machine code (sometimes called object code), with contractual agreements prohibiting the client from reverse-compiling it to yield equivalent code in a more easily analyzed, higher-level language like Fortran.[29]

---

[26] For instance, case law has sometimes treated the term "mathematical algorithm" as synonymous with a "mathematical formula" such ss sin(2a) = 2(sin(a)cos(a)) or a "law of nature" such as E = (mass) (speed of light squared). (For discussions see: Donald S. Chisum, "The Patentability of Algorithms," *University of Pittsburgh Law Review, vol. 47, No. 4,* summer 1986, pp. 959-992; and Supreme Court cases *Gottschalk v. Benson, 409* U.S. 63 (1972), *Parker v. Flook,* 437 U.S. 584 (1978), and *Diamond* v. *Diehr, 450* U.S. 381 (1981).)

[27] For a computer scientist's perspective on legal confusions resulting from unsuitable models for algorithms and computer programs, see Allen Newell, "The Models Are Broken, The Models Are Broken!" *University of Pittsburgh Law Review, vol. 47, No. 4,* summer 1986, pp. 1023-1035.

Some think that new and useful algorithms, including "mathematical" algorithms should constitute subject matter eligible for patent protection (see Donald S. Chisum, ibid., pp. 959-1022.)

A recent decision by the Court of Appeals for the Federal Circuit (*In re Iwahashi, et al.,* CAFC 89-1019, decided Nov. 7, 1989) reversed PTO's rejection of a patent application in which the algorithm constituted the bulk of the invention. Some observers consider that this decision, which limited patent protection for the algorithm only to ita use in the particular apparatus described in the claims, will further ease the way for patent protection for algorithms. (See Edmund L. Andrews, "Patents: Algorithm Ruling May Aid Software," *The New York* Times, Nov. 11,1989, business section, p. 36.)

[28] By contrast, patent rights are granted in exchange for full disclosure of the patentee's "beat method" of practicing the invention. But some consider that in practice claims are sometimes so broad that they don't really show the state-of-the-art (B. Kahin, personal communication, Dec. 1, 1989),

[29] Absent contractual agreements, trade secret does not protect against reverse engineering.

[30] The Copyright Office has special release provisions for deposit of software with trade secrets. Depositors may use object code, remove trade-secret pare% etc.

PC software, by contrast is mass-marketed to hundreds of thousands of customers.[30] To help maintain trade-secret status, PC software is often published in object code and distributed with a "shrink-wrap" license, which every purchaser is supposed to agree to on opening the package. The shrink-wrap license may contain language and terms that purport to create a duty by the user to maintain the trade-secret information, but some question whether this would stand up in court.[31]

[30] The Copyright Office has special release provisions for deposit of software with trade secrets. Depositors may use object code, remove trade-secret parts, etc.

[31] See Anne W. Branscomb, "Who Owns Creativity?" *Technology Review*, May/June 1988, p. 43., and also Sherman et al., op. cit., footnote 2, sec. 309.4(g). Anne Branscomb believes that statutory clarification is needed for the status of trade secrets within copyrighted works (Anne W. Branscomb, personal communication, Dec. 8, 1989).