July 2015

The Protection of Property Rights in Computer Software

Edward W. Rilee

Please take a moment to share how this work helps you through this survey. Your feedback will be important as we plan further development of our repository.

Follow this and additional works at: https://ideaexchange.uakron.edu/akronlawreview

Part of the Computer Law Commons, Intellectual Property Law Commons, and the Property Law and Real Estate Commons

Recommended Citation

Available at: https://ideaexchange.uakron.edu/akronlawreview/vol14/iss1/7

This Article is brought to you for free and open access by Akron Law Journals at IdeaExchange@UAkron, the institutional repository of The University of Akron in Akron, Ohio, USA. It has been accepted for inclusion in Akron Law Review by an authorized administrator of IdeaExchange@UAkron. For more information, please contact mjons@uakron.edu, uapress@uakron.edu.
THE PROTECTION OF PROPERTY RIGHTS
IN COMPUTER SOFTWARE

I. INTRODUCTION

During the last decade a number of attempts have been made by the courts in the realm of patent and copyright law to settle the issue of the protection of property rights in computer software. These traditional methods of protection, however, have not been able to assimilate this relatively new technological invention. Likewise, at the start of a new decade, little or no progress towards a comprehensive form of software protection can be detected. This paper will examine the problems associated with using federal patent or copyright law to provide computer software protection and discuss why state trade secret protection has remained the primary guarantee against infringement of property rights in computer software.

A. Definition of Terms

Before discussing the advantages and disadvantages of each of the primary forms of protection of computer software, it is necessary to define the computer-related terms that will be utilized in the remainder of this article.

Computer "software" may simply be defined as everything except computer "hardware." Hardware is the physical electrical circuits, motors, tape drives, relays, printers and readers, and memory which make up the mechanical and electrical parts of the computer. In contrast to computer hardware which has little useful application without programming, computer software instructs the various hardware components of the computer. Software may be characterized as falling within one of two major categories: 1) operation or system software or 2) application software.

System software is composed of the software which is usually resident within the computer memory and remains part of it during all operations. It performs the scheduling and translating of the application software and

---

1 For example, the question of the patentability of computer software has come before the United States Supreme Court on three separate occasions without a complete resolution of the issue. See Parker v. Flook, 247 U.S. 584 (1978); Dann v. Johnston, 425 U.S. 219 (1976); Gottschalk v. Benson, 409 U.S. 63 (1972). A discussion of these cases appears in the text accompanying notes 50-84.


maintains general control of the computer functions. System software, generally written in "machine language," is a low-level computer language that is unintelligible except to the computer. This type of language most closely resembles the mathematical equivalent of a series of "on/off" switches. Machine language and system software is unique to an individual computer model and would not function on another computer of a different design. However, system software does provide the means to translate high-level applications software into a set of instructions which can operate the computer hardware.

Application software is what is commonly referred to as a computer program. This is what an individual user will write and place in a computer to produce a desired result. Application software is written in a high-level computer language. For example, a few of the most widely used languages include FORTRAN (FORmula TRANslation), COBOL (COMmon Business Oriented Language), BASIC, and PL/1. These languages are human-oriented and use common words or mathematical symbols, such as "Add A to B" or "C = A + B," to perform different operations. Each computer system software translates these universal, high-level languages into the machine language the individual computer can understand. Application software, hereafter referred to as a computer program, may be composed of one or more mathematical expressions called "algorithms." An algorithm is simply a mathematical solution to a problem (e.g., "if A is greater than B, add B to C"). It must be written exactly in the form of the high-level computer language being used, such as BASIC, or the computer will not perform the functional intent of the programmer.

B. The Development of a Computer Program

Traditionally, there have been two steps in the development of a computer program. First, a programmer would devise a "flow chart" before writing the computer program in a chosen high-level language. A flow chart is merely a graphical outline of what the programmer intends the computer program to do, such as add the month's invoices, calculate depreciation or estimate product life. Today, many programmers no longer prepare a flow

---

5 Id.
7 Ross, supra note 3, at 755 n.140.
8 Id.

---
chart on paper but have a mental image of what is desired and proceed directly to the second step in computer programming, writing the "source" program.  

The source program is composed of a series of algorithms written in a high-level language which the programmer has chosen to accomplish his goal. It should be remembered, however, that a computer can only perform what it is programmed to do. The source program may simply be written on paper, punched on cards or paper tape, or stored on magnetic tape or computer memory. As a result, any computer whose system software has been designed to translate the high-level language of the source program into machine language that the computer can understand will be able to use the source program.

The source program is then "read" into the computer. This may be accomplished in one of two ways. One method is by direct communication with the computer by means of a typewriter-type keyboard. Or the source program can be recorded on magnetic or paper tape and the tape then transferred to the computer. As the computer reads the source program it transforms it, by means of its system software, into a series of electrical impulses. Finally, the computer translates the electrical representation of the source program into a machine language "object" program which the computer is able to understand. This latter program may also be printed by the computer on cards, paper tape, or magnetic tape for use on similar computers. Source and object programs are usually the products which are sold, licensed, disclosed, and, on occasion, misappropriated. It is for these forms of computer software, the source and object application programs, that it is most difficult to offer meaningful protection of the developer's property rights.

II. AN OVERVIEW OF PROPERTY RIGHT PROTECTION IN THE COMPUTER SOFTWARE FIELD

The need for protection of property rights in computer software has not always been recognized. Early computer programs were either for government or university-supported research programs and had little im-

---

12 Generally, source programs are what programmers commonly refer to as computer programs.
13 Id.
14 The court in Data Cash noted: "Object programs, which enter into the mechanical process itself, cannot be read without the aid of special equipment and cannot be understood by even the most experienced programmers." 480 F. Supp. at 1065.
15 See, e.g., Comment, Copyright Protection for Computer Programs Under the 1976 Copyright Act, 52 Ind. L.J. 503, 506 (1977).
mediate commercial application. The programs were individualized for particular machine requirements and, as a result, were immune from piracy. Furthermore, as the computer became more commercial many computer manufacturers introduced the marketing concept of “bundling.” Bundling meant that the user bought the computer and received the software free. This was important to the infant computer industry since many potential users did not possess in-house computer expertise. However, an independent software industry emerged late in the 1950’s and early in the 1960’s. This fact, coupled with the availability of in-house programmers, caused manufacturers to stop providing free computer software systems. It was the emergence of these independent software companies that expanded the need for effective computer software protection.

Any method of protection should adequately protect the financial investment made by the software developer and allow him to attain a reasonable profit. In addition, the public should benefit because the developer will be encouraged to invest in additional software programs if the risks of misappropriation are minimized. Consequently, technological progress would be encouraged by dissemination of significant computer software developments. These goals often appear to be mutually exclusive, however, since any increase in availability of software in which a developer has a property right results in a corresponding increase in the risk that the developer’s property rights will be compromised.

Three primary forms of protection have been utilized for the protection of property rights in computer software: 1) patent, 2) copyright, and 3) trade secrecy. Throughout the subsequent discussion of these areas, it is clear that one difficulty in providing protection for computer software emanates from its nature. For example, computer software is obviously a writing, but this writing can take many physical and electrical forms. Likewise, the content of the writing may be the embodiment of a mathematical formula, a set of instructions to control another machine, or a unique computational method. Understanding the nature of the content of the software, as well as the form it is in, is the first step in deciding which forms of protection will best protect the property interests of the developer. Furthermore, apparent and sometimes real inconsistencies in court

---

17 Id.
18 Id.
20 See M. Pope & P. Pope, supra note 4, at 527. Generally, these commentators feel that these methods have had varying degrees of success or failure.
rulings can best be understood and distinguished by determining how the court views the application of the computer software and the protection the court affords that application.22

III. PATENT PROTECTION OF COMPUTER SOFTWARE

A patent is a federal grant of exclusive monopoly for controlling who may make, use, or sell an invention.23 This grant of monopoly is valid against independent discovery24 and encourages an inventor to disseminate his invention by guaranteeing him a right to civil action and damages for infringement.25 The patent provides a definite term of protection, seventeen years, after which the invention becomes the property of the public.26 To be patented the invention must meet strict standards of proper subject matter, novelty, and nonobviousness.27 Patent law will only protect inventions which consist of a "new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof."28 As a result, patents will not be issued for discovery of laws of nature, mathematical expressions (algorithms), or other fundamental truths.29 A mere idea without an end-use is not patentable.30

The first patent applications for computer programs were based upon machine claims31 and premised on the logic that a new computer program intended for a general purpose computer gives rise to a patentable invention.32 The courts initially upheld these patent claims for computer software.33 In 1966, however, a Report by the President's Commission on the Patent

22 See e.g., authorities cited note 2 supra.
23 Section 271(a) of the United States Code provides, "Except as otherwise provided in this title, whoever without authority makes, uses or sells any patented invention, within the United States during the term of the patent therefor, infringes the patent." 35 U.S.C. § 271(a) (1976).
24 Id.
26 Section 154 of the United States Code provides, "Every patent shall contain a short title of the invention and a grant to the patentee, his heirs or assigns, for the term of seventeen years, subject to the payment of issue fees as provided for in this title, of the right to exclude others from making, using, or selling the invention throughout the United States, referring to the specification for the particulars thereof." 35 U.S.C. § 154 (1976).
28 Id. § 101.
31 See, e.g., M. Pope & P. Pope, supra note 4, at 536.
32 Id. at n.43.
33 See In re Foster, 438 F.2d 1011 (C.C.P.A. 1971); In re Musgrave, 431 F.2d 882 (C.C.P.A. 1970); In re Mahony, 421 F.2d 742 (C.C.P.A. 1970); In re Bernhart, 417 F.2d 1395 (C.C.P.A. 1969); In re Prater, 415 F.2d 1393 (C.C.P.A. 1969).
System recommended that patents not be issued for computer software. Prior to this report, the Patent Office had given notice of proposed new guidelines for patent application. Published in 1968, these guidelines maintained that computer software was no longer patentable whether based on machine or process claims. This view followed the rationale of Cochrane v. Deener which stated that a process was only patentable if a "physical transformation" occurred in a given article. As such, computer programs were only patentable if the computer was transformed into a special purpose machine by the computer program. This approach followed the "mental steps" doctrine set out in In re Abrams which held that if the process could be carried out with pencil and paper it was unpatentable.

Shortly after the publication of the guidelines, In re Prater was decided. The Patent Office had rejected Prater's machine and process claims because they were tied to a general purpose computer. However, the Court of Custom and Patent Appeals (CCPA) rejected the Patent Office arguments. Upon rehearing, it affirmed the machine claim but rejected the process claims for lack of specification. The court reaffirmed the approach it had originally taken in Ex Parte King and, shortly thereafter, the Patent Office withdrew its guidelines.

In accordance with the CCPA's rejection of the Patent Office's approach to the patentability of computer software, four computer software patents issued by the Patent Office were appealed and reversed by the CCPA. In the patent claims case of In re Barnhart, the court reaffirmed

36 94 U.S. 780 (1876).
37 The Supreme Court in Cochrane stated: "A process is a mode of treatment of certain materials to produce a given result. It is an act, or a series of acts, performed upon the subject-matter to be transformed and reduced to a different state or thing." Id. at 788.
38 See M. Pope & P. Pope, supra note 4, at 537.
39 188 F.2d 165 (C.C.P.A. 1951).
41 Id. at 415 F.2d 1393, 1406. In order for a patent to be granted, the inventor must provide a complete description of the apparatus or process. Section 112 of the United States Code provides, "The specification shall contain a written description of the invention, and of the manner and process of making and using it, in such full, clear, concise, and exact terms as to enable any person skilled in the art to which it pertains, or with which it is most clearly connected to make and use the same, and shall set forth the best mode contemplated by the inventor of carrying out his invention." 35 U.S.C. § 112 (1976).
44 See Scafetta, supra note 9, at 392, n.45.
and noted the differences between the computer and the "mental steps" of the human mind. In In re Musgrave, however, the CCPA departed from the "mental steps" test of In re Abram and promulgated a new test. The court held that the mere fact that some steps can be carried out "mentally" does not preclude acceptance as long as the invention is considered to be a "technological art" as defined by Section 101. According to the court, the "technological arts" include those arts that aid the constitutional purpose of promoting the "useful arts." The CCPA went on to reverse two more computer software patent applications in favor of the inventors by applying the "technological arts" test of In re Musgrave.

The first case dealing with the question of computer software patents to reach the Supreme Court was Gottschalk v. Benson. The Benson invention was a computational algorithm which converted binary coded decimal (BCD) numbers into ordinary decimal numbers. The Patent Office, maintaining its restricted view of the patentability of computer software, rejected the patent claims. The CCPA, following its approach in In re Musgrave, reversed the Patent Office and found that the claims passed the scrutiny of Section 101's "useful arts" test, thus affirming its rejection of the In re Abram "mental steps" doctrine. On review, the United States Supreme Court reversed the CCPA decision and held that the claims could not be allowed since they would effectively preempt the use of the algorithm for any other use.

While the Benson decision was widely seen as barring patents for all computer software, the Supreme Court expressly stated that its decision did not preclude future patent protection of computer software. In subse-
sequent decisions the CCPA has failed to grasp this aspect of the *Benson* decision by holding that patent claims directed toward mathematical results are unpatentable. 57

Since *Benson*, the CCPA has continued to decide the validity of patent claims for software on the basis of whether the process is directed toward a mathematical (unpatentable) or non-mathematical (patentable) result. 58 Nevertheless, the CCPA’s narrow interpretation of *Benson* clashed with the Patent and Trademark Office’s (PTO) view. 59 As a result, when the second computer software patentability case, *Dann v. Johnston*, 60 came before the United States Supreme Court, it was hoped the Court would settle the confusion surrounding the *Benson* decision. 61

**Dann v. Johnston** involved a “machine system for automatic record keeping for bank checks and deposits.” 62 Actually, the system was a general purpose computer and computer program. 63 The Patent Office had rejected the application for a patent, but the CCPA reversed 64 on the grounds that the machine-type claims did not come within the process restricted holding of *Benson*, nor was the claim in the nature of an algorithm which could not be patented. 65 Rather than expanding *Benson* to exclude machine as well as process claims or limiting *Benson* by merely denying patent protection to the algorithms, the Court reversed the CCPA decision on the ground of obviousness under Section 103. 66

Following **Dann v. Johnston**, the CCPA continued to utilize the mathematical/non-mathematical distinction enunciated in *Benson* to determine the patentability of computer software. 67 For example, in **In re Freeman** 68

---

57 See, e.g., **In re Waldbaum**, 559 F.2d 611, 612 (C.C.P.A. 1977) (the patent application had been previously upheld by the C.C.P.A. but was reversed on rehearing); **In re Christensen**, 478 F.2d 1392 (C.C.P.A. 1973); see also, **In re Waldbaum**, 457 F.2d 997 (C.C.P.A. 1972).

58 See generally, Nimtz, supra note 16, at 15, nn.65, 66 and accompanying text.

59 Id.


61 See note 55 supra. The Patent Office interpreted *Benson* as effectively preempting all patents for computer programs; however, the C.C.P.A. limited *Benson* as only applying to those computer programs which were directed solely to a mathematical result.


63 Id. at 222.

64 **In re Johnston**, 502 F.2d 765 (C.C.P.A. 1974).

65 Id. at 771.

66 425 U.S. at 220, 225-230. Section 103 of the United States Code provides, “A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negative by the manner in which the invention was made.” 35 U.S.C. § 103 (1976).

67 See note 58 supra,
the CCPA limited Benson as applying only to mathematical algorithms and thus allowed patents on other non-Benson computer software. Consequently, an attempt was made by the Patent Office to get more cases before the United States Supreme Court. In Parker v. Flook, the invention was a method for updating alarm limits in processes involving catalytic conversion of hydrocarbons. The claim involved an algorithm designed to calculate a revised alarm limit and then adjust this limit to its new value. The Patent Office rejected Flook's claims, citing the CCPA's decision in In re Christensen which held that additional steps added to an otherwise unpatentable method could not render the material patentable. The CCPA reversed the Patent Office and restricted Christensen to factual situations where nothing actually occurs after the algorithm.

While noting that Flook limited his claims to the use of the algorithm only for catalytic conversion of hydrocarbons, the United States Supreme Court reversed the CCPA holding. The Court saw the issue in Flook as whether the discovery and application of a novel algorithm would enable an otherwise non-patentable method of control to be patentable. It held that it would not but once again reserved the right to uphold software patents. The Court also maintained that a claim is not unpatentable merely because it contains an algorithm but, instead of utilizing the Benson test of whether the patent would preempt other uses of the algorithm, the Court stated that a process will be patentable if the process itself, not merely the algorithm, is new and useful.

Following Flook, the CCPA has restricted its application by continuing to allow patent applications for computer software. On the other hand, the
Patent Office has broadly interpreted Flook in denying similar patent applications.83 Thus, Flook has not resulted in a clarification of the question of the patentability of computer software but has only maintained the status quo. As a result, the parameters of patent protection remain obscure eight years after the Benson decision.

IV. COPYRIGHT PROTECTION OF COMPUTER SOFTWARE

A copyright is a limited monopoly which protects the originality of a work which has been properly published, registered with the Copyright Office, and affixed with a copyright notice.84 Under the 1976 copyright law, the protection extends for the life of the author plus fifty years.85 In the event that the author has been hired to produce the work, which would be the case with most computer software, the protection of the copyright extends for a period of 75 years from original publication or 100 years from original creation of the work.86

Copyright protection may be obtained for a published computer program by attaching a copyright notice to the computer program, delivering two copies of the program to the Register of Copyrights, and paying the filing fee.87 While a copyright will protect the holder from infringement by exact or near exact copies, it offers little protection against the incorporation of the program concepts in another form. Similar programs using the same mathematical concepts do not constitute a copyright infringement.88 This limitation on the scope of copyright protection is based on a distinction between the physical form of a published work and the use of the concepts it contains.89

In Baker v. Seldon80 the United States Supreme Court refused to extend copyright protection so far as to prevent utilization of a method published in a copyrighted work. Sheldon had written a book outlining a new bookkeeping system and had obtained a copyright on the book. It contained suggested forms to implement the bookkeeping system. Baker began publishing forms which, while different than Seldon's, accomplished the same end. The Supreme Court held that Seldon's copyright did not give him the exclusive right to make and use any bookkeeping forms utilizing the new

83 Sarkar, 588 F.2d at 1331; Johnson, 589 F.2d at 1081.
86 Id.
87 17 U.S.C. §§ 401(b), 408(b), 708 (1976).
90 101 U.S. 99 (1879).
bookkeeping system but only those forms which were copyrighted as part of his book.\(^9\)

Almost thirty years after *Baker v. Seldon* the Supreme Court decided *White-Smith Music Publishing Co. v. Appollo Co.*\(^9\) This case has become a prime example of the extent to which protection will be granted to computer software under the copyright law. Appollo Company manufactured pianola rolls for player pianos which, when played, reproduced music which had been copyrighted by White-Smith. The Supreme Court refused to recognize an infringement of White-Smith’s copyright.\(^9\) Instead, the Court held that the pianola roll was not a “copy” of White-Smith’s music as the word was intended under the law.\(^9\) Although the Copyright Act of 1909 extended copyright protection to the pianola roll situation,\(^9\) *White-Smith* has continued to stand for the principle that, in order to be a copy, the form must be such as to be perceptible by the human eye.

The Register of Copyrights first began to accept computer software for registration in 1964.\(^9\) The Copyright Office viewed the question of computer software registration as revolving around two central questions:

1. Whether a program is the “writing of an author” and, thus, copyrightable, and
2. Whether a reproduction of the program in a form actually used to operate or be “read” by a machine is a “copy” that can be accepted for copyright registration.\(^9\)

The Copyright Office indicated that it would continue to resolve doubtful issues in favor of computer software copyright protection.\(^9\) One commentator felt that there was little doubt that any new copyright statute would expressly provide for computer software and reject the narrow definition of “copy” set out in *White-Smith*.\(^9\)

The Copyright Act of 1976, the first complete revision since the 1909 Act, did not meet these expectations. In fact, Section 117 of the 1976 Act expressly states:

[T]his title [title 17] does not afford to the owner of copyright in a work

---

\(^{91}\) *Id.* at 107.
\(^{92}\) 209 U.S. 1 (1908).
\(^{93}\) *Id.* at 18.
\(^{94}\) *Id.* at 17.
\(^{96}\) See Announcement from the Copyright Office, *Copyright Registration for Computer Programs*, 11 BULL. COPYRIGHT SOC’Y U.S.A. 361 (1964).
\(^{97}\) *Id.*
\(^{98}\) *Id.*
any greater or lesser rights with respect to the use of the work in conjunction with automatic systems capable of storing, processing, retrieving, or transferring information, or in conjunction with any similar device, machine or process, than those afforded to work under the law, whether title 17 or the common law or statutes of a State, in effect on December 31, 1977, as held applicable and construed by a court in an action brought under this title.\textsuperscript{100}

Thus, the effect of the Copyright Act of 1976 has been to maintain the \textit{status quo} regarding the copyright protection of computer software.\textsuperscript{101}

Yet, while clearly not wishing to become involved in the problems of extending the copyright protection of computer software at this time, Congress has left the language in the 1976 Act broad enough to accept the various forms of computer software in the future. For example, Section 101 of the Act defines “copies” as:

Material objects, other than phonorecords, in which a work is fixed by any method now known or later developed, and from which the work can be perceived, reproduced, or otherwise communicated, either directly or with the aid of a machine or device.\textsuperscript{102}

The Act of 1976 also extends copyright protection to unpublished material if a copyright notice is affixed.\textsuperscript{103} This protection is exclusively within the Act.\textsuperscript{104} Breach of licensing agreements, however, is expressly preempted from coverage\textsuperscript{105} with the result that disclosures by an employee of in-house computer software to competition would not be considered a copyright infringement.\textsuperscript{106}

Recognizing the difficulties associated with providing adequate copyright protection for computer software, Congress created the National Commission on New Technological Uses of Copyrighted Works (CONTU) in 1974 to make specific recommendations for legislation.\textsuperscript{107} Initial guidelines promulgated by the Commission state that adequate protection of computer software under the copyright law should: 1) prevent unauthorized copying of computer programs, 2) facilitate the lawful use of computer programs, 3) aid the development and dissemination of computer programs, and 4) avoid granting more economic power than is necessary to provide

\begin{thebibliography}{1}
\bibitem{100} 17 U.S.C. § 117 (1976).
\bibitem{103} Id. §§ 102, 103, and 104.
\bibitem{104} Id. § 301.
\bibitem{105} Id. § 301(b).
\bibitem{106} See Comment, \textit{supra} note 15, at 507, n.29.
\end{thebibliography}
an incentive to develop computer programs. The Commission's final report, released in 1978, recommended that Congress: 1) amend Section 101 to expressly provide for computer programs to be copyrighted, 2) repeal Section 117 which maintains the status quo, and 3) redraft Section 117 to allow lawful users of computer programs to utilize or adapt these programs for their own purposes. The Commission was not unified in its recommendations; a strong dissent questioned whether computer software was a proper subject for copyright protection. In addition, the dissent recommended that separate legislation be enacted to provide protection of computer software property rights which would, in essence, recognize its unique nature.

Enactment of specific legislation or modification of the Copyright Act of 1976 has not yet occurred and problem areas remain in utilizing the copyright to protect property rights in computer software. For example, a copyright will only protect against substantial and unauthorized copying. In addition, unauthorized use of the concepts expressed in the computer software is not protected. This means a unique algorithm which is part of a copyrighted computer program may be utilized in a different form by another without constituting infringement of the copyrighted program itself. Furthermore, courts have not been willing to view an object deck as a copy of a copyrighted computer software source deck. As long as a useful version (e.g., object deck) of a computer program can be created which is not a "copy" within the meaning of the copyright law, the copyright will provide little protection of the developer's property rights.

V. TRADE SECRET PROTECTION OF COMPUTER SOFTWARE

Trade secrecy is the most widely used form of protection of property rights in computer software and is derived from the common law and

---

108 NATIONAL COMMISSION ON NEW TECHNOLOGICAL USES OF COPYRIGHTED WORKS, FINAL REPORT (July 31, 1978).
109 17 U.S.C. § 101 (1976) (this section of title 17 merely defines the terms used within the title).
111 See notes 95-100 supra and accompanying text.
112 NATIONAL COMMISSION ON NEW TECHNOLOGICAL USES OF COPYRIGHTED WORKS, FINAL REPORT 2 (July 31, 1978).
113 Id. at 69.
114 SOFTWARE SUBCOMMITTEE OF THE NATIONAL COMMISSION ON NEW TECHNOLOGICAL USES OF COPYRIGHTED WORKS, REPORT (March 5, 1977).
115 See generally, M. Pope & P. Pope, supra note 4, at 546.
117 For example, in the Data Cash case, the court held that even if the plaintiff's object program had been copied by the defendant, it was not within the legal definition of "copy" and therefore was not actionable. 480 F. Supp. 1063, 1064 (N.D. Tex. 1979).
A trade secret is defined as a "formula, pattern, device or compilation of information which is used in one's business, and which gives him an opportunity to obtain an advantage over competitors who do not know or use it."120

There are numerous advantages in acquiring trade secret protection. For example, it is not necessary to register computer software as a trade secret in order to obtain legal protection for its development. Also, a computer trade secret does not have to meet the rigid standards of subject matter, novelty, and non-obviousness which are necessary to obtain a patent.121 Even if a computer program would otherwise be patentable, the developer may choose to elect trade secret protection rather than protection under patent law since the right to patent protection will be lost if the application is not made within one year after the software has been used commercially.122

Unlike patent or copyright protection which have definite terms, trade secret protection may be maintained indefinitely. The protection provided under state trade secret laws may be lost, however, if the computer software is published or otherwise becomes known.123 The developer seeking protection under the laws of trade secrecy must be able to show that adequate precautions were taken to maintain the confidentiality of the computer software.124 Courts have placed this burden of proof of adequate precaution upon the party seeking damages or an injunction under the trade secret laws.125 Trade secrecy also does not offer protection against independent development of the computer software.126

Legal action is normally brought under trade secret laws based on one of two theories: 1) misappropriation, or 2) breach of contract, either by unlawful disclosure by an employee or by violation of a licensing agreement.127 As discussed above, the burden of proof is on the moving party to show adequate steps have been taken to protect his trade secret.128 Limited access areas and employee non-disclosure contracts may be used

119 See Restatement of Torts § 757 (1939).
120 Id. at Comment b.
123 See, e.g., Bender, supra note 118, at 928.
125 Id. at 348.
126 See Restatement of Torts § 757 (1939).
128 See notes 124-125 supra.
to reduce the risk of loss of the trade secret by misappropriation or wrongful disclosure. Since computer software is often licensed for use by another party, the licensor should be careful to deal with responsible licensees. If he does not, the right to protection under state trade secret laws may be lost since courts are apt to view this as a failure to take adequate protective measures.\footnote{Id.}{129}

Other than the difficulties of administering and maintaining computer software secrecy, federal preemption has been a major roadblock to the effective use of state trade secret laws for the protection of computer software. The preemption of state law by federal law may occur when: 1) federal and state law are in direct conflict, 2) Congress intended to exclude state regulation, or 3) the burden of similar dual state and federal regulation is considered too great.\footnote{A discussion of the federal preemption doctrine may be found in The Federalist No. 32, p. 241 (B. Wright ed. 1961), cited in Goldstein v. California, 412 U.S. 546, 553 (1972). For a detailed discussion of the application and history of the federal preemption doctrine, see 412 U.S. at 552-560.}{130} Five major cases decided by the United States Supreme Court have been instrumental in defining what will or will not be allowed protection under state trade secret law.

The first two of these cases, Sears, Roebuck & Co. v. Stiffel Co.\footnote{376 U.S. 225 (1964).}{131} and Compco Corp. v. Day-Brite Lighting, Inc.,\footnote{376 U.S. 234 (1964).}{132} involved lighting fixtures which had been patented. An action was brought under both federal patent and state unfair competition laws. The patents were struck down by the lower courts but the unfair competition action for damages and injunction was affirmed.\footnote{Sears, Roebuck & Co. v. Stiffel Co., 313 F.2d 115 (7th Cir. 1963); Compco Corp. v. Day-Brite Lighting, Inc., 311 F.2d 26 (7th Cir. 1962).}{133} The Supreme Court reversed both decisions.\footnote{376 U.S. at 232, 233; 376 U.S. at 238, 239.}{134} The Court in Sears held that to grant plaintiff's invention patent-like protection under state law when no federal patent was warranted would be to give greater protection than the federal patent law provided.\footnote{See 376 U.S. at 232.}{135} The Court refused to allow such an anomaly. While the Sears and Compco cases were not brought before the Court under state trade secret laws, commentators saw the cases as severely limiting availability of state trade secret protection for unpatentable inventions.\footnote{Id. note 13 and accompanying text. For example, one commentator noted: [In the Sears-Compco opinions, Justice Black does not confine himself to this narrow definition of public domain. Instead, he implies that all inventions are initially in the public domain regardless of whether a patent was sought covering them. Carried to the limit of its logic, this would seem to say that all state trade secret protection for inventions is preempted. Adelman & Jaress, Inventions and the Law of Trade Secrets after Lear v. Adkins, 16 Wayne L. Rev. 77, 82 (1969).}{136}
The third major case to come before the Supreme Court was *Lear, Inc. v. Adkins.* Adkins had contracted with Lear to develop an improved gyroscope for aircraft use. Adkins confidentially disclosed his method to Lear in return for royalty payments and agreed to license the invention under any forthcoming patent. Lear stopped paying Adkins royalties on the ground that nothing new had been invented. A patent was subsequently issued to Adkins and he sued Lear for both pre-patent and post-patent royalties under state trade secret and federal patent laws, respectively.

Lear contended that no royalties were due for the post-patent period because the patent was invalid. Adkins argued that the doctrine of licensor-licensee estoppel prevented a patent licensee from challenging the validity of the licensor's patent. The Supreme Court overruled the licensor-licensee estoppel doctrine and held that Adkins would not be able to bring an action for the patent royalties until the issue of the validity of the patent was decided. The Court thus held that public disclosure of the invention by way of the patent precluded Adkins from recovering royalties under state trade secret laws. The Supreme Court then refused to decide whether, and to what extent, the states may act to protect unpatented trade secrets and remanded the case to the lower court to determine what pre-patent award royalties, if any, were due.

The dissenters in *Lear,* Justices Black, Douglas, and Chief Justice Warren, argued that *Sears* and *Compco* had already decided that state trade secret laws were preempted by federal patent law. In addition, they viewed federal patent law as preempting state trade secret laws both prior and subsequent to disclosure. Lower courts have had some difficulty with this decision and the dissent’s view of *Sears* and *Compco.* As a result, one court distinguished *Sears* and *Compco* as applying only to inventions which were applicable directly to the public and not to private licensing agreements.

The fourth major case to come before the Supreme Court breathed new life back into the protection of property rights by state trade secret

---

138 Id. at 659.
140 Id. at 676.
141 Id. at 668.
142 Id. at 675-676.
143 Id. at 676-677.
laws. Goldstein v. California was a criminal action brought under a California statute prohibiting phonorecord piracy. The defendant argued that the statute was preempted by federal copyright law which did not allow the copyrighting of phonorecords at the time the piracy took place. The Supreme Court upheld the California statute, noting that the Constitution did not expressly vest all power to grant copyright protection in the federal government and that the California statute did not directly conflict with federal copyright law. The Supreme Court also distinguished and limited its early holdings in Sears and Compco to situations in which state law conflicted with the objectives of federal patent law.

The issue of federal preemption of state trade secret laws was finally put to rest in Kewanee Oil Co. v. Bicron Corp. The Court of Appeals for the Sixth Circuit seemed to follow the dissent in Lear and held that state trade secret laws were preempted by federal patent law if the invention would have been patentable, but a patent had not been applied for or the inventor had lost the right to apply for a patent through commercial use of the invention for more than one year. Many commentators disagreed and regarded the decision as the beginning of the end of state trade secret laws. The Supreme Court, however, reversed and held that no present or real possibility of future conflict existed between federal patent and state trade secret laws. The Court stated, “Certainly the patent policy of encouraging invention is not disturbed by the existence of another form of incentive to invention. In this respect the two systems [patent and trade secret law] are not and never would be in conflict.”

The apparent inconsistency between Lear and Kewanee is best explained in Aronson v. Quick Point Pencil Co. Aronson involved a contract for the use of an invention not yet patented. Royalties of 5% of the selling price were to be paid if a patent was issued and 2½% if a patent was not granted. Quick Point argued that the contract for the royalties was preempted by federal patent law since Aronson failed to receive a patent. The Supreme Court rejected this position and held that federal patent law

---

147 412 U.S. 546 (1972).
149 412 U.S. at 558.
150 Id. at 559.
151 Id. at 569-570.
154 Id. at 1086.
155 See, e.g., Wydick, supra note 127, at 757.
156 416 U.S. at 493.
157 Id. at 484.
does not preempt state contract law so as to preclude enforcement of the contract. The Court viewed the payment of the 2½% royalty as the price Quick Point was willing to pay to enable it to advance in the marketplace. The Court then distinguished Lear as not applying when a patent had never been issued.

VI. CONCLUSION

Today, federal copyright law only protects the copyright holder from infringement by exact or near exact copies. The copyright does not provide protection against the incorporation of computer program concepts in another form. Recent cases tend to question the applicability of patents regarding the protection of computer software and the issues not answered by the Supreme Court in Benson still prevent drawing clear guidelines for the patentability of computer software. In addition, Lear may preempt concurrent state protection in the event a computer software patent previously issued is later held invalid.

Nevertheless, despite the fact that state trade secret laws limit free disclosure, do not offer protection against independent discovery, and may require licensing agreements and restrictive covenants to prevent the loss of the trade secret to competitors, the United States Supreme Court in Kewanee affirmed the right of the states to provide independent protection of intellectual property rights. This decision has opened the way for effective protection of computer software under state trade secret laws. Compared to the inadequacies of copyright safeguards and the uncertainties of patent restraints, state trade secret law offers the best, albeit limited, protection.

EDWARD W. RILEE

159 Id.
160 Id. at 264.
161 The Court in Aronson said, “In Lear . . . we held that a person licensed to use a patent may challenge the validity of the patent, and that a licensee who establishes that the patent is invalid need not pay the royalties accrued under the licensing agreement subsequent to the issuance of the patent. Both holdings relied on desirability of encouraging licensees to challenge the validity of patents, to further the strong federal policy that only inventions which meet the rigorous requirements of patentability shall be withdrawn from the public domain. . . . Accordingly, neither the holding nor the rationale of Lear controls when no patent has issued, and no ideas have been withdrawn from public use.” Id.