October 1996

Wobbling on the Shoulders of Giants: The Supreme Court's Failure in Lotus v. Borland

Bradley W. Grout

Follow this and additional works at: https://digitalcommons.law.uga.edu/jipl

Part of the Constitutional Law Commons, and the Intellectual Property Law Commons

Recommended Citation
Available at: https://digitalcommons.law.uga.edu/jipl/vol4/iss1/5

This Notes is brought to you for free and open access by Digital Commons @ Georgia Law. It has been accepted for inclusion in Journal of Intellectual Property Law by an authorized editor of Digital Commons @ Georgia Law. Please share how you have benefited from this access. For more information, please contact tstriepe@uga.edu.
NOTES

WOBBLING ON THE SHOULDERS OF GIANTS: THE SUPREME COURT’S FAILURE IN LOTUS V. BORLAND

“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 Right to their respective Writings and Discoveries.”¹ Out of this constitutional power came the federal copyright² and patent³ laws granting limited proprietary rights to those who hold a valid copyright or patent. While the distinction between the two forms of intellectual property rights is generally obvious,⁴ the creation of computer software has blurred that distinction significantly.

Computer software is both expensive and useful. Its unique, dual nature of expression and usefulness has caused great confusion throughout the legal community. The underlying written computer code for software provides the expressive aspect because of its inherent literary nature. Certain words and symbols carry significance that any author could appreciate. This code, however, is also a useful process; it causes the computer to perform specific functions. Because both of these elements are present in software,

¹ U.S. CONST. art. I, § 8, cl. 8.
⁴ Copyrights generally apply to an “expression” of a particular idea, but not to the idea itself or a process behind the idea. 17 U.S.C. § 102(b) (1994). Patents, however, protect the process or invention as long as it is new and useful. 35 U.S.C. § 101 (1994). In most conventional cases, courts have no trouble deciding if patent or copyright applies to the subject matter because the difference between a useful process or invention is clearly separate from the expression of an idea.
the creator can now obtain both copyright and patent protection for his software.  

As the use and ownership of computers increased, and as programs became more complex, many theories emerged on the proper form of protection that should be afforded to software. Copyright was the original method of protection, but its scope was relatively limited at the outset. Not until the landmark case of *Apple Computer, Inc. v. Franklin Computer Corp.* did the scope of copyright protection for computer software begin to expand. Some commentators now argue that patent law, not copyright law, is the most effective protection for software and should be the sole form of protection. Although the courts originally declined to apply patent law to computer software, they have increasingly upheld software patents since the Supreme Court's decision in *Diamond v. Diehr*. 

As the courts and scholars continue to debate over which of the two traditional forms of intellectual property is best suited to protect computer software, a proposal for a new type of intellectual property protection has become popular. This new form would be a legal hybrid designed to reach a compromise between copyright and patent law. It would combine various elements of both

---

5 See Willis E. Higgins, *Technological Poetry: The Interface Between Copyright and Patents For Software*, 12 HASTINGS COMM. & ENT. L. J. 67 (1989) (explaining cooperative protection provided by both copyright and patent and arguing that a combination of these two traditional forms of property rights is sufficient protection such that a new legislative right is unnecessary). See infra notes 10-11 and accompanying text for a brief discussion of the new right.


8 See, e.g., Gottschalk v. Benson, 409 U.S. 63, 175 U.S.P.Q. (BNA) 673 (1972) (landmark case holding computer programs to be little more than standard mathematical formulas (algorithms) and therefore not protectable under patent laws, as patent laws did not cover laws of nature or mathematical formulas).


traditional forms of intellectual property in order to protect the unique, dual nature of computer software. This *sui generis* system would be the sole protection necessary for software and would render the copyright-patent debate moot.

The Supreme Court recently had a golden opportunity to render obsolete much of the debate over which form of intellectual property should govern software. When it granted certiorari from the First Circuit to hear the controversial *Lotus Development Corp. v. Borland International, Inc.*\(^\text{12}\) many lawyers, professors, developers, and the like, in the intellectual property field, eagerly awaited a clarifying decision.\(^\text{13}\) The decision, however, left most sorely disappointed. In a split opinion with one justice abstaining, the Court merely affirmed the First Circuit holding\(^\text{14}\) with a one-sentence bench decision.\(^\text{15}\) The Court provided no policy, no reasoning, and no guidance as to the proper role of copyright in the realm of software protection.\(^\text{16}\)

This Note will argue that the Supreme Court should have denied copyright protection to Lotus’s software and that it should have laid down affirmative principles to establish a proper role for copyright in the software industry. Also, the Note will explain the ramifications of the Court’s failure to provide guidance for future software protection. Part I will provide an historical development of intellectual property as applied to software. Part II will analyze

\(^{11}\) *Sui generis* literally means "of its own kind or class; i.e., the only one of its own kind." *BLACK’S LAW DICTIONARY* 1434 (6th ed. 1990). The *sui generis* right would have to be created by the legislature, not the courts.


\(^{16}\) *Id.* The opinion simply stated, “[t]he judgment of the United States Court of Appeals for the First Circuit is affirmed by an equally divided Court.” *Id.*
the First Circuit's decision denying copyright protection and explain the failure of the Supreme Court to resolve the issue itself. Part III will argue that the Court correctly affirmed the First Circuit, but that it also should have clarified the ambiguities left by the lower court's opinion by setting a workable definition of copyrightable expression. Finally, Part IV will detail the possible effect of the Court's failure to render an effective precedent.

I. HISTORICAL BACKGROUND OF SOFTWARE PROTECTION

The power of Congress to issue copyrights, as noted earlier, comes from Article I of the Constitution. Congress acted on this constitutional grant in establishing the first Copyright Act in 1790. Although the first Act was extremely limited in scope, subsequent Acts continued to expand on what a valid copyright could protect.

The landmark case of *Baker v. Selden* explains the difference between an idea, which is not copyrightable, and the expression of that idea, which is copyrightable. Selden was the owner of a registered copyright on his book, *Selden's Condensed Ledger, or Book-keeping Simplified*. This book introduced a new method of accounting, including drawings and sketches that accompanied and explained the textual expression. Selden also obtained copy-
rights for all improvements and additions to the book in an attempt to protect his new system.24 When Baker published his own accounting books using substantially the same ledger system that Selden introduced, Selden filed suit alleging an infringement on his copyright.25

The court ruled that Selden could protect his expression of the idea so that no one else could reproduce his books of explanation, but such protection did not extend to the method of accounting itself:26 "[W]hilst no one has a right to print or publish his book, or any material part thereof . . . any person may practise and use the art itself which he has described and illustrated therein."27 The Court stated that letters-patent would be the only potential form of protection (if any were allowed at all), but did not examine a patent claim.28

Courts have long struggled with this idea-expression dichotomy as it applies to computer software—and for good reason. For, as will be demonstrated, the line between an idea and its expression is a very hard one to draw in the world of complex computer software.

A. THE SCOPE OF COPYRIGHT PROTECTION FOR COMPUTER SOFTWARE

The question as to how to protect computer software was relatively insignificant until the advent of personal microcomputers and mass-produced software in the 1970s.29 In 1976, Congress enacted a new Copyright Act giving protection to "original works of

24 Id.
25 Id. at 100-101.
27 Id. at 104. To see how this early policy applies to software, note the discussion by the Lotus I court, laid out in part II.B., infra.
28 Baker, 101 U.S. at 105.
29 Vance Franklin Brown, Comment, The Incompatibility of Copyright and Computer Software: An Economic Evaluation and a Proposal for a Marketplace Solution, 66 N.C. L. REV. 977, 979-80 (1988). Even during the increase in computer usage throughout the 1970s, the question over how to protect software was easy to answer: use copyright. "[M]ost of the software community viewed copyright as the only form of intellectual property protection available for software." Higgins, supra note 5, at 74 (citing Committee Reports, 1986 A.B.A. SEC PAT., TRADEMARK & COPYRIGHT L. 262). Thus copyright was the dominant protection used by software developers.
authorship fixed in any tangible medium of expression." The Act provides protection where works can be "perceived, reproduced, or otherwise communicated, either directly or with the aid of a machine or device." However, subsection (b) of the Act limits the protection to exclude "any idea, procedure, process, system, [or] method of operation . . . ."

While the Act itself says nothing about computer software, the legislative history shows that the Act was intended to extend coverage to the relatively new medium. Congress was indeed concerned that copyright, under the description in subsection (a), might be applied to the processes within a program, so it included subsection (b) in an attempt to prevent such an extended scope of protection.

Congress added two basic changes to the Act of 1976. These changes, recommended by the National Commission on New Technological Uses of Copyrighted Works (CONTU), were

---


31 17 U.S.C. § 102 (1994) (emphasis added). The emphasized language would suggest that a computer program, as long as it qualifies as a literary work (per § 102(a)(1)) is clearly copyrightable. Most courts have ruled that a software program is a sort of literary work because of its written code. See, e.g., Lotus I, supra note 14, at 817; Apple Computer, Inc. v. Franklin Computer Corp., 714 F.2d 1240 (3d Cir. 1983), cert. dismissed, 464 U.S. 1033 (1984).


33 H.R. Rep. No. 94-1476, 94th Cong., 2d Sess. 57 (1976), reprinted in 1976 U.S.C.C.A.N. 5659. "[N]ew expressive forms—electronic music, filmstrips, and computer programs, for example—could be regarded as an extension of copyrightable subject matter Congress had already intended to protect, and were thus considered copyrightable from the outset without the need of new legislation." Id. at 5664.

34 Id. at 5670. Subsection (b) of the Act was included to ensure that the methodology of or used in running the program would not be afforded copyright protection. Only the literal writing of the code would receive protection under the Act, consistent with prior law (namely Baker v. Selden). That code is the "literary" element subject to protection.

35 CONTU was created in 1974 to address and respond to concerns about the impact of newer technology on existing copyright laws. One of the goals of this commission was to recommend any specific legislation that would adequately protect software developers. Brown, supra note 29, at 984. However, one commissioner dissented from the recommendations where a program "is capable of being used to control computer operations." NATIONAL COMM’N ON NEW TECH. USES OF COPYRIGHTED WORKS, FINAL REPORT 2 (1978) (reprinted in 3 COMPUTER L. J. 53 (1981)) [hereinafter "CONTU Final Report"]. Because most programs do control computer operations, this dissent demonstrates exactly the problems courts have in attempting to apply copyright law. The First Circuit, in its Lotus I decision, appears to rely on this dissent as the basis for its decision. See infra note 206 and accompanying text.

Even with this dissent and CONTU’s acknowledgment of the difficulty, the fact that
adopted in the 1980 Amendments\textsuperscript{36} in order to clarify the scope of the 1976 Act.\textsuperscript{37} The first change adds a definition of computer program: “A ‘computer program’ is a set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result.”\textsuperscript{38} The second change, which is far less significant for purposes of this Note, allows an owner of a copy of copyrighted software to make copies or changes for his own use.\textsuperscript{39}

These two changes were based on the following four policy statements:

(1) Copyright should proscribe the unauthorized copying of these works.
(2) Copyright should in no way inhibit the rightful use of these works.
(3) Copyright should not block the development and dissemination of these works.
(4) Copyright should not grant anyone more economic power than is necessary to achieve the incentive to create.\textsuperscript{40}

While the language of the statute clearly seems to protect only the literal expression of a program, distinguishing between the program’s expression and its underlying processes, methods of

\textsuperscript{36} This was called the Computer Software Copyright Act. Brown, supra note 29, at 984.


\textsuperscript{39} 17 U.S.C. § 117 (1994). This section adds express approval for reverse engineering and other fair use capabilities in connection with Judge Boudin’s dissenting opinion in \textit{Lotus I}, discussed at part II.E., infra.

operation, and systems proved difficult for the courts. At the outset, courts followed the legislative mandate by separating the functional from the merely expressive aspects of computer software. Courts would not allow copyright to protect any functional elements. If a function contained even some expressive elements, the functional aspect precluded copyright protection for the expressive parts of that function.

Apple Computer, Inc. v. Franklin Computer Corp. marked the beginning of the expansion of copyright protection. Since that decision, courts have gradually expanded the scope of copyright protection beyond a program's code. In addition to protection against the copying of a program's underlying code, copyright began to offer protection against the nonliteral "copying of a program's overall structure, sequence, organization, and even its manner of operation."

In 1986, the Third Circuit, in Whelan Associates, Inc. v. Jaslow Dental Laboratory, Inc., handed down a highly controversial opinion concerning the copyrightability of computer software.

---

41 See Synercom Tech., Inc. v. University Computing Co., 462 F. Supp. 1003, 199 U.S.P.Q. (BNA) 537 (N.D. Tex. 1978) (holding that expression that cannot be separated from its underlying idea is not copyrightable). The judge in this case used an analogy to the "figure-H" configuration of a gear box in an automobile. Judge Higginbotham reasoned that any expression of the design, such as a manual, diagram, etc., is protectable through copyright. However, the designer may not prevent a manufacturer from utilizing the figure-H in its car, or from using any variation thereof. Society should be able to benefit from the new design and build upon it for constant improvement. Id. at 1013.

42 "An expression through which a user controls the underlying process is not protected by copyright, regardless of the number of ways the control can be expressed." Ramos, supra note 40, at 272 (analyzing figure-H analogy from Synercom, 462 F. Supp. at 1013).

43 714 F.2d 1240 (3d Cir. 1983), cert. dismissed, 464 U.S. 1033 (1984) (holding that computer operating system programs are not necessarily precluded from copyright protection). Franklin copied the Apple II object code so that its ACE 100 model would be compatible with Apple programs and compete with Apple in general—the ACE 100 could then run the same programs.

That object code made up the program that actually ran the computer and caused it to perform whatever functions the software "told" it to perform. Object code is the "machine language" in binary code, distinct from the source code that computer programmers use to write the software. Source code is much easier for humans to read and is much easier to classify as expression.

44 Brown, supra note 29, at 986.

45 Id. at 987.

The allegedly infringing program in that case did not "copy" any elements of the original's code. Instead, the overall structure of the program was the same as the original. The court determined that the new program nevertheless violated the copyright on the original software. 47

The Whelan court ruled that the purpose of the program constitutes the idea behind it. 48 Copyright protection can then extend to all elements of the program that are "not necessary to [that] purpose." 49 The court's definition of expression included "the manner in which the program operates, controls and regulates the computer in receiving, assembling, calculating, retaining, correlating, and producing useful information either on a screen, print-out or by audio communication." 50

This new view of what constitutes copyrightable expression marked the extension of copyright to the "look and feel" 51 of a computer program. Such expansion meant that not only would a program that substantially copies an original’s code violate a copyright, but any program that uses the same (or substantially similar) format, operation, or structure as an original would violate that copyright. 52

The decision in Whelan seems contrary to the holding in Synercom Technology, Inc. v. University Computing Co., where the court ruled that the sequencing and ordering of data for input

---

47 Id. A software developer created a custom program to keep records for a dental laboratory. The lab then developed a similar overall program in a different program language and then began to market the new version on its own.

48 Id. at 1236.

49 Id. The reasoning must follow that if the purpose constitutes an idea, anything that is not absolutely necessary to the purpose would simply be an expression of the idea. The expression itself can be protected by copyright, but the ides cannot.

50 Id. at 1239. As pointed out in Brown, supra note 29, at 988, this definition seems to ignore 17 U.S.C. § 102(b). That section forbids copyright protection to processes and methods of operation. In allowing copyright to protect the second program in Whelan, the Third Circuit focused too much on rewarding the developer for his efforts. While that is one consideration underlying copyright policy, the more important consideration is promoting the public good. That is best achieved by allowing programmers to build on each other's work. This building policy is the major theme underlying this Note's position against copyright for Lotus 1-2-3 in Lotus v. Borland.

51 Brown, supra note 29, at 995 (citing Davidson, The Whelan Decision: Missing the Middle Ground, 5 COMPUTER L. REP. 335, 337 (1986); Russo and Derwin, Copyright in the "Look and Feel" of Computer Software, COMPUTER LAW, Feb. 1985, at 1).

52 Brown, supra note 29, at 993.
formats were inseparable from the underlying idea or purpose.\textsuperscript{53} The \textit{Whelan} court attempted to distinguish \textit{Synercom} on the basis that the dispute in the latter case only involved input formats (which are essential to the program) while the former was concerned with the structure and sequence of the program in its entirety.\textsuperscript{54}

\textit{Broderbund Software, Inc. v. Unison World, Inc.}\textsuperscript{55} was decided later in the same year as \textit{Whelan} and involved virtually the same situation as presented in \textit{Synercom}. In \textit{Broderbund}, however, input instructions appeared on the screen, whereas the input formats in \textit{Synercom} were in a printed manual. Each set of input instructions served the same purpose—both helped the user enter data properly so that the programs could perform their respective functions.\textsuperscript{56} Even though the case was factually similar to \textit{Synercom}, the \textit{Broderbund} court found copyright infringement because the programs were substantially similar. Adopting the reasoning of \textit{Whelan}, the \textit{Broderbund} court held that a software developer “must express the idea [of its rival program] through a substantially different structure.”\textsuperscript{57}

The decision in \textit{Lotus Development Corp. v. Paperback Software International}\textsuperscript{58} marked the height of broad copyright protection.\textsuperscript{59} That case involved facts very similar to \textit{Lotus I}: another spreadsheet program, “VP-Planner,” was created to be fully compatible with Lotus 1-2-3 so that no retraining or rewriting of macros would

\textsuperscript{53} \textit{Synercom}, 462 F. Supp. at 1013. That court posed the rhetorical question: “If sequencing and ordering is expression, what separable idea is expressed?” \textit{Id}.  

\textsuperscript{54} 797 F.2d at 1239. The difference must lie in that the overall structure is merely an expression and arrangement of the idea, whereas the sequencing and ordering of the input formats are an inherent part of the idea and not just an expression of it. The distinction is a weak one and has caused confusion in later cases, as many courts decided not to adopt the \textit{Whelan} analysis. \textit{See}, e.g., Plains Cotton Coop. Assn. v. Goodpasture Computer Serv., Inc., 807 F.2d 1256, 1262, 1 U.S.P.Q.2d (BNA) 1635 (5th Cir.), \textit{cert. denied}, 484 U.S. 821 (1987) (expressly declining to follow \textit{Whelan}).  


\textsuperscript{56} \textit{Id}. at 1132.  

\textsuperscript{57} \textit{Id}. at 1133.  


\textsuperscript{59} \textit{See} Ramos, supra note 40, at 267 (“[V]irtually every aspect of a computer process is [now] a copyrightable expression.”).
be necessary. Paperback developed this new software without using Lotus code, but fully intended to copy the basic structure of the user interface. Judge Keeton held the user interface to be protectable by copyright. This development marked a further expansion from Whelan. In Paperback, the court prohibited programs that produce a similar function as an original, even where the program does not copy the original's underlying, written code.

Finally, the Second Circuit, in Computer Associates International v. Altai, began to restrict the newly expanded scope of copyright protection. It did so by introducing a test to determine substantial similarity between a copyrighted program and its alleged infringer. The court broke down its analysis into three steps: abstraction, filtration, and comparison. Abstraction means breaking the original program down into its various structural parts. Filtration requires a determination of which parts are protectable as expression and which are inseparable from the idea expressed.

---

60 Paperback, 740 F. Supp. at 69. For a definition of "macro" and a factual statement of the Lotus I case, see infra, part II.A.
61 Id.
62 Id. at 68. Paperback never attempted to hide the fact that its interface was identical to Lotus's. Rather, it used that fact to its own advantage as part of its advertising and boasted that VP-Planner operates the same way as Lotus 1-2-3 so that the user would not have to learn anything new to do all the same things. VP-Planner just cost a lot less. Id. at 69.
63 Id. at 706-11. This three-step analysis has been warmly received by courts and commentators. See, e.g., Engineering Dynamics, Inc. v. Structural Software, Inc., 26 F.3d 1335, 31 U.S.P.Q.2d (BNA) 1641 (5th Cir. 1994); Apple Computer, Inc. v. Microsoft Corp., 35 F.3d 1435, 32 U.S.P.Q.2d (BNA) 1086 (9th Cir. 1994), cert. denied, 115 S. Ct. 1176 (1995); Gates Rubber Co. v. Bando Chemical Indus., 9 F.3d 823, 28 U.S.P.Q.2d (BNA) 1503 (10th Cir. 1993); Atari Games Corp. v. Nintendo of America, Inc., 975 F.2d 832, 24 U.S.P.Q.2d (BNA) 1015 (Fed Cir. 1992). Even the Lotus I court acknowledged the appropriateness of the test given the right context. Supra note 14, at 814.
64 Altai, 982 F.2d at 706.
65 Id. at 707.
Comparison then involves comparing those protectable parts to the allegedly infringing program. If there is substantial similarity between significant protected elements, the latter program infringes on the original's copyright. Applying this test, the Altai court affirmed the district court's finding of no infringement and emphasized that "copyright infringement cases . . . that involve computer programs are highly fact specific." 70

Even with the general acceptance of the Altai test, courts still vary in their application of that test as well as in their application of other copyright policies. 71 Unfortunately, none of the many available tests truly seems to fit the wide array of programs on the market. This multiplicity of judicial approaches testifies to the confusion that still surrounds the copyrightability of computer programs. 72 To clear up this confusion, the Supreme Court needed to set some clear and authoritative guidelines for determining the proper scope of copyright protection for computer software. Because the Court failed to take advantage of the opportunity in Lotus v. Borland, copyright doctrine in this area remains as murky as ever.

B. THE RISE OF PATENTABILITY

Patents were initially rejected for computer software, primarily because of the judiciary's misunderstanding about what a computer program really is. 73 Patents have never been permitted to protect "laws of nature, mathematical formulas, abstract ideas, or other fundamental truths." 74 The Supreme Court applied this

---

68 Id. at 710.
69 Id.
70 Id. at 715.
71 Such other policies include merger, scenes à faire and fair use. All of these will be discussed in Part III of this Note.
72 Pamela Samuelson et al., A Manifesto Concerning the Legal Protection of Computer Programs, 94 COLUM. L. REV. 2308, 2349 n.150 (1994) (citing Whelan, Lotus I, Paperback, and Gates as decisions offering different tests that could all apply, but that are not sufficiently tailored to suit software needs).
73 Griem, supra note 7, at 147, 163-67. The chief misconception was that a computer program is nothing more than a combination of mathematical formulas. By following this line of thought, judges "completely discounted the painstaking human effort and the patentable innovation that went into creating the conversion algorithm." Id. at 164.
74 Brown, supra note 29, at 981.
principle in *Gottschalk v. Benson*\(^75\) to deny patent protection for a computer program.\(^76\) The court held that an algorithm is a type of mathematical formula, which is not patentable,\(^77\) and, since the computer program is based on a series of algorithms, the program was not patentable either.\(^78\)

In 1981, the Court revisited the issuance of patents for computer software after gaining a more sophisticated view of the subject matter. The landmark decision of *Diamond v. Diehr*\(^79\) finally permitted patent law to encompass software. The Court recognized that programs are not simply algorithms standing alone (which, like any other mathematical formula, would not be patentable); rather, they are inventions that utilize algorithms to perform tasks designed by the programmer.\(^80\) The requirements for patent protection are met "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 the patent laws were designed to protect."\(^81\) Although initially hesitant about issuing software patents,\(^82\) the Patent and Trademark Office has gradually become more comfortable with its authority over software.\(^83\) Many intellectual property experts strongly advocate patent rather than copyright protection for computer software because of its closer fit to the utilitarian nature of software.\(^84\)

---

\(^76\) *Id.* at 71-72.
\(^77\) *Id.* at 72.
\(^78\) *Id.* at 72.
\(^79\) 450 U.S. 175, 209 U.S.P.Q. (BNA) 1 (1981). It is important to note that *Diamond* did not overrule *Gottschalk*. The *Gottschalk* Court simply held that the particular program at issue was not patentable and specifically left the door open for future patent protection of computer software. *Gottschalk*, 409 U.S. at 71.
\(^80\) *Id.* at 192-93.
\(^81\) *Id.* at 192.
\(^84\) *See*, e.g., Griem, *supra* note 7; Higgins, *supra* note 5.
C. THE CONFLICT BETWEEN PATENT AND COPYRIGHT PROTECTION

Because patent law is designed to protect the very utilitarian nature of programs that copyright law cannot address, the fact that both are used for the same medium inevitably leads to conflict between the two areas of law. The scope and use of copyright protection for software seemed to expand while patents were still uncommon. Copyright offered a quicker and easier solution than patent law because patents were far more difficult to obtain. Despite the hardships encountered in obtaining a patent, such protection was still an attractive option because, "in exchange for the difficulty in obtaining patent protection . . ., patent law grants a monopoly." Such monopolistic power can motivate developers to apply for patents more frequently than they otherwise would.

Patents eventually gained acceptance in the courts and with the PTO. Copyright, however, maintained its presence in the software industry, and an overlap developed between the two forms of protection. Theoretically, such an overlap should not occur due to the distinct nature of each type of intellectual property: patents protect useful inventions, whereas copyrights only protect expressions of those ideas. As courts recognized this conflict and the inherent usefulness of even the most expressive programs, they began to restrict the scope of copyright to its pre-Whelan level of

---

86 Copyrights are relatively easy to obtain through a simple, inexpensive registration process. Higgins, supra note 5, at 69. The person seeking a copyright for his work need only submit an application, 17 U.S.C. § 408(a) (1994), and a copy of the work to be protected, 17 U.S.C. § 408(b)(1) (1994).

Patents, on the other hand, require a much more tedious and expensive process. More specific requirements must be met with respect to originality, 35 U.S.C. § 102, and obviousness, 35 U.S.C. § 103. The total process (as of 1987) generally takes about two or three years to complete. Higgins, supra note 5, at 70-71 (citing Comm'r. of Patents and Trademark Ann. Rep. at 21 (1987)).

87 If a developer can obtain a monopoly over a certain market, then he could profit handsomely. That would make the effort and expense of the application process for patents worthwhile. Id.

88 See part I.B., supra.

89 See discussions, supra parts I.A. and I.B.
LOTUS V. BORLAND

protection.\(^9\) Still, the newly restricted scope was unacceptable because, as discussed earlier, courts did not apply the law uniformly.\(^9\)

The decision in *Lotus v. Borland* could have proven monumental by finally defining a limited scope of copyright protection for computer software. Such a limiting decision would have allowed subsequent expansion of patent protection for software or might have opened the door for the development of a *sui generis* system of protection of computer programs. The Supreme Court could have returned the law to a state where copyright protection once again applied to nothing in the software but the actual code or other purely expressive feature of the software. The First Circuit took a big step in that direction by recognizing that even the expressive aspects of software may be uncopyrightable as methods of operation. The Supreme Court, however, failed to seize the opportunity to establish some firm guidance on the proper scope of copyright protection as applied to computer software.

II. LOTUS V. BORLAND: THE CASE AND DECISIONS

A. FACTS

Lotus had been the market leader\(^9\) in computer spreadsheet programs with its popular software, Lotus 1-2-3.\(^9\) A user operates the program by using commands designed to perform specific functions (print, save, etc.).\(^9\) Each of these commands is contained in a menu or submenu such that the user need only select the command from the menu directly with the cursor keys or

---

\(^9\) Ramos, *supra* note 40, at 274 (citing Plains Cotton Coop. Ass'n v. Goodpasture Computer Serv., Inc., 807 F.2d 1256 (5th Cir.), *cert. denied*, 484 U.S. 821 (1987), which declined to follow *Whelan* and others that modified the *Whelan* rule).

\(^9\) Courts did not apply rules consistently. Some applied the *Whelan-Broderbund* policy and others modified them according to individual cases. *See, e.g.*, Engineering Dynamics, Inc. v. Structural Software, Inc., 26 F.3d 1335 (5th Cir. 1994) (holding user interface is copyrightable only to extent that it reflects non-functional judgment). *But see* Lotus Dev. Corp. v. Paperback Software Int'l, 704 F. Supp. 57 (D. Mass. 1990) (allowing copyright for user interface as whole).

\(^9\) Microsoft Excel is now the undisputed leader in spreadsheets.

\(^9\) Lotus did have a valid registered copyright for protection of Lotus 1-2-3.

\(^9\) *Lotus I*, *supra* note 14, at 809.
mouse, or type a corresponding letter as indicated in the menu. Rather than go through each menu and submenu with different commands individually to perform one final function, a user can write a "macro" that will combine a series of commands into one keystroke for faster operation. Macros essentially allow the user to customize the program for his own efficiency.

Borland is a competitor of Lotus with its own spreadsheet software, Quattro and Quattro Pro. These two programs were designed to be far superior to Lotus 1-2-3 with many functions that were not available on Lotus's program. In order to attract users who were already accustomed to using Lotus 1-2-3, Borland included an option in Quattro and Quattro Pro that displayed "a virtually identical copy of the entire 1-2-3 menu tree." In providing this display that looked like the Lotus menu structure, Borland used its own code, not simply a copy of the Lotus code. Importantly, however, Borland did copy exactly the observable words and structure of Lotus 1-2-3 so that Lotus 1-2-3 users would be able to use Quattro or Quattro Pro without having to learn a new system of menus and commands. Borland even titled this Quattro feature the "Lotus Emulation Interface."

Because the command hierarchy was the same, Borland also enabled Quattro to read previously written Lotus macros. Not only would users save time by operating the software based on the same

---

95 Jd.

96 Jd. Many programs allow users to write macros to simplify use of the software. They are generally for functions that a user performs often. For example, word processor users may create a macro for certain letterheads to start different documents so that one action brings up a particular heading rather than having to type and format from scratch every time.

97 Jd. at 810 (quoting Judge Keeton in Lotus Dev. Corp. v. Borland Int'l, Inc., 831 F. Supp. 202, 212, 30 U.S.P.Q.2d (BNA) 1081 (D. Mass 1993)). This "menu tree" is the menu command hierarchy that is the focus of the court's opinion. By "command hierarchy," the court is referring to the structure of the menus and commands within the menu. For example, if a menu were to contain the commands "copy," "save," and "print," with each one of those commands calling up a submenu, and so on, that would be a command hierarchy.

98 Borland's code is considered by most to be superior to the Lotus code. It runs the program more efficiently, among other things, and is one of the reasons that Quattro Pro is the better software.

99 Lotus I, supra note 14, at 810. This feature was removed after the district court's summary judgment in favor of Lotus, but the commands and structure remained the same, simply without the visual reproduction of Lotus 1-2-3. Quattro maintained compatibility with Lotus 1-2-3 and could still read and run macros written for Lotus software.
system, they would not even have to rewrite their macros. Quattro does offer many other features that Lotus does not, and users do not have to use the Lotus interface if they do not choose to do so. A Quattro user who does not operate on Lotus commands may use Quattro's own commands and command structure.

Lotus filed suit against Borland only four days after it won its decision in *Paperback*. There the District Court of Massachusetts held the user interface to be copyrightable expression because it involved a choice of words and arrangement of those words on the screen. The district court, following its ruling in *Paperback*, found the Lotus menu command hierarchy to be copyrightable expression. Therefore, in copying that command hierarchy, which was an integral part of the Lotus 1-2-3 software, Borland infringed Lotus's valid copyright as a matter of law.

The First Circuit disagreed. The court ruled that the menu command hierarchy is not merely an expression, regardless of the test applied to determine whether there was literal copying of the Lotus program. The command structure as a whole cannot be copyrighted in the first place because it is a method of operation. Thus the First Circuit reversed Judge Keeton's decision based on the prohibition against copyright protection for a method of operation as set out in 17 U.S.C. § 102(b).

B. THE FIRST CIRCUIT'S ANALYSIS

Borland asserted in its defense that the menu command hierarchy is exactly the sort of uncopyrightable method of operation

---


101 The lower court held that "[a] very satisfactory spreadsheet menu tree can be constructed using different commands and a different command structure from those of Lotus 1-2-3. In fact, Borland has constructed just such an alternate tree for use in Quattro Pro's native mode. Even if one holds the arrangement of menu commands constant, it is possible to generate literally millions of satisfactory menu trees by varying the menu commands employed." *Lotus I*, supra note 14, at 810-11 (quoting district court, *Lotus Dev. Corp. v. Borland Int'l*, Inc., 799 F. Supp. 203, 217 (D. Mass. 1992)).


103 *Id.* at 815.

104 *Id.*

105 *Id.* at 819.
intended to be excluded under the Copyright Act of 1976.106 Because Borland conceded that it had copied the command structure, the only issue for the court was whether or not that structure could be classified as an "expression" or as a "method of operation."107 Because Paperback was a district court decision, it had no precedential value to the Court of Appeals and left the Lotus I court "navigating in uncharted waters."108

One key distinction that separated Lotus I from other cases concerning copyright protection for software was that it dealt with a different issue. In any copyright infringement suit, a plaintiff must satisfy a two-prong test originally established by Justice O'Connor in Feist: "(1) ownership of a valid copyright, and (2) copying of constituent elements of the work that are original."109 Whereas most prior software copyright cases assumed a valid copyright and simply addressed the question of copying, the First Circuit focused on the valid copyright prong of the Feist test.110 The First Circuit was thus one of the first courts to challenge the presumption that copyright could protect any expressive element of computer programs.111

In finding that Lotus's copyright was invalid, the First Circuit first concentrated on separating method from expression. The court

---

106 Id. at 815 (referring to 17 U.S.C § 102(b)).
107 Lotus I, supra note 14, at 812.
109 Feist, 499 U.S. at 361.
111 The court considered using the Altai test, but stated that focusing on copying would keep the "real" issue in the background. In this type of case, the court should turn its attention toward a determination of whether the work is even copyrightable from the outset, not whether another program might infringe. Thus it is an issue for the court, not for the jury. Lotus I, supra note 14, at 815.
did not deny that the menu command hierarchy is expressive; rather, the court determined that it is a method of operation for the program that forces the otherwise protectable expression into the realm of § 102(b).\textsuperscript{112} Thus, even though the menu command hierarchy is expressive and would normally be protectable, the fact that it is also a method of operation renders it unprotected by copyright law.

In finding that the command hierarchy is a method of operation, the First Circuit employed an analogy to the buttons (and labels on the buttons) of a video cassette recorder (VCR).\textsuperscript{113} Even though the buttons of a VCR might arguably be expressive in the way they are laid out on the face of the machine, or in the choice of labels on them, they are still necessary for a user to operate the VCR.\textsuperscript{114} If the user wishes to play a cassette, he must press the button marked "play." Similarly, if a Lotus user wishes to print a document, he must highlight the "Print" command or type the letter "P."\textsuperscript{115} That is the computer equivalent of pushing a VCR button.\textsuperscript{116}

The commands themselves are the essential part of the Lotus 1-2-3 software and entirely different from the labels on a VCR's buttons.\textsuperscript{117} The court emphasized this point because the labels themselves may be a copyrightable expression. On a VCR, the labels for the buttons merely provide guidance to the user by indicating which buttons to press. Even if unlabeled, though, the buttons would still operate the VCR. Thus, the buttons are the

\textsuperscript{112} 17 U.S.C. § 102(b) (1994). The court elaborated on this point, explaining that "[t]he 'expressive' choices of what to name the command terms and how to arrange them do not magically change the uncopyrightable menu command hierarchy into copyrightable subject matter." \textit{Lotus I}, supra note 14, at 816.

\textsuperscript{113} \textit{Id.} at 817. In doing so, the court rejected Borland's attempt to analogize the case to \textit{Baker v. Selden}. \textit{Id.} at 814. Borland compared the T-accounts of Selden's accounting system to the command structure of Lotus 1-2-3. Its argument was that both were input functions (Selden's on paper, Lotus's on the screen). The difference, according to the First Circuit, was that Lotus merely claimed protection over its \textit{expressive} choice of commands, not over the entire structure of spreadsheet programs in general (Selden had claimed protection for the general structure of his T-account system).

\textsuperscript{114} \textit{Id.} at 817.

\textsuperscript{115} \textit{Id.}

\textsuperscript{116} \textit{Lotus I}, supra note 14, at 817.

\textsuperscript{117} \textit{Id.}
functional part of the VCR while the labels are purely expressive.\textsuperscript{118}

Conversely, a user could not perform a computer equivalent of pressing the button without a command.\textsuperscript{119} The command itself is more than an expression of a function; it is the function (the command operates like the actual button, not just the label). In other words, the expressive element is inseparable from the functional element. The court indicated that if Lotus had somehow designed its software with the terms on the interface acting merely as labels, such terms might have been copyrightable under traditional software copyright analysis.\textsuperscript{120} However, the commands are not mere labels—the exact terms are necessary to perform the desired functions. Removing the command would remove the function, unlike removing the label from the button of a VCR.\textsuperscript{121} Software commands serve the same purpose as the VCR's actual buttons: operating the software. As such, the commands are part of the structure's method of operation.\textsuperscript{122}

In addition to using the VCR analogy to deny copyright coverage to Lotus's interface, the First Circuit also adhered to the policies behind copyright protection.\textsuperscript{123} According to the court, the Copyright Act should "encourage others to build freely upon the ideas and information conveyed by a work."\textsuperscript{124} To hold that Borland had infringed a valid Lotus copyright in this case would run contrary to that policy and Congressional intent behind § 102(b). Borland, and others, would not be able to use Lotus’s ideas to create better, more sophisticated and efficient software. By denying this ability to build on a previous developer’s ideas, copyright would not truly “promote the Progress of Science and

\textsuperscript{118} \textit{Id.}
\textsuperscript{119} \textit{Id.}
\textsuperscript{120} See, \textit{id.} While the court did not say this directly, it made a significant effort to distinguish such a case. Because the visual terms would then merely be guides rather than essential to the operation of the software, they would no longer fall under the method-of-operation exception. Hence, under the \textit{Paperback} user interface analysis, a court may find the “labeling” terms to be protectable by copyright.
\textsuperscript{121} \textit{Lotus I, supra} note 14, at 817.
\textsuperscript{122} \textit{Id.}
\textsuperscript{123} \textit{Id.} at 818.
\textsuperscript{124} \textit{Id.} (quoting \textit{Feist Publications, Inc. v. Rural Tel. Serv. Co.}, 499 U.S. 340, 350 (1991)).
useful Arts." The First Circuit thus acknowledged that most successful programs derive from programmers who stand "on the shoulders of giants."

C. THE UNITED STATES SUPREME COURT'S DECISION

When the Supreme Court granted certiorari, many intellectual property professors and practitioners from around the country hoped for a monumental decision. *Lotus v. Borland* would represent the first case of its kind before the Court. Many people had already seen the First Circuit decision as a potential (and actual) landmark case to refine the role of copyright in the ever-increasing world of computer usage. In a single case, *Lotus* presented many of the controversial issues of copyright law with respect to computer software. Even if the Court were to decide not to follow the First Circuit’s analysis, at least it could settle the confusion over the proper role of copyright protection. Unfortunately, the Supreme Court disappointed everyone with its decision.

On January 16, 1996, only months after deciding to hear the case on certiorari, the opinion came down: "[t]he judgment of the United States Court of Appeals for the First Circuit is affirmed by an equally divided Court... Justice Stevens took no part in the consideration or decision of this case." No explanation accompanied this message, nor did any of the Justices write separately. The First Circuit opinion simply stood as it was.

---

125 *Lotus I*, supra note 14, at 818 (quoting U.S. CONST. art. I, § 8, cl. 8).
126 This phrase comes from Isaac Newton’s famous statement: "If I have seen farther than other men, it is because I have stood on the shoulders of giants." When applied to computer programming, this refers to "building" on prior works. Programmers build on the creations of those who come before them. These original creators are the giants who provide the boost for those who can improve the creation.
127 Only circuit courts and Congress have addressed the copyrightability of computer software. The Supreme Court has only addressed software in the patent context as discussed in part I.B., supra.
128 Many of the top academics and lawyers in the United States prepared and submitted an Amicus Curiae brief to the Court in favor of a restricted role for copyright. Amicus Brief, supra note 13.
129 Namely, the case involved the focus on the first prong of the *Feist* test, a rejection of copyright based on § 102(b), a sound means of determining a method of operation, and a rejection of the *Altai* Abstraction-Filtration-Comparison test.
Perhaps the only good part of the holding was simply a happen-
stance of the four-four split: affirming the judgment in favor of
Borland. While the menu command hierarchy (and user interface
in general) should not be copyrightable, many reasons exist to
explain why. The Supreme Court’s failure to issue a full decision
leaves the circuits in disarray over the proper resolution of several
important issues regarding the copyrightability of computer
software.

Indeed, this Note has already discussed some of those unsettled
issues. When should a court apply the *Altai* test (or a variation
thereof)? The First Circuit applies it only in cases of nonliteral
copying, but other circuits attempt to apply the test even in cases
of literal copying. What can be copyrighted at all? How functional
must an expressive element be before copyright will no longer
protect the work? Is the First Circuit’s VCR analogy helpful in
determining whether other aspects of computer software fall under
§ 102(b)? The Supreme Court left all of these questions unan-
swered. The balance of this Note will demonstrate that *Lotus v.
Borland* presented the Court with the opportunity to resolve
several important issues. It will also explain how the Court should
have decided, or at least set up guiding principles for deciding, each
issue.

III. MISSED OPPORTUNITIES

Although the First Circuit’s main holding in *Lotus I* simply
declared a menu command hierarchy an uncopyrightable method
of operation, the case also raised many of the controversial issues
that confront software copyright cases. The fact that so many
issues were present in a single case made it perfect for a Supreme
Court decision that addressed each one. Indeed, the Court could
have set forth proper tests for copyrightability, where necessary,
and eliminated the use of doctrines that should not apply to
software.

The First Circuit raised five issues in its opinion that the
Supreme Court could have addressed. First, the First Circuit
distinguished cases that merely separate ideas from expression and
declared that copyright may not cover even some expressive
elements when those elements are also methods of operation.\textsuperscript{131} Second, the First Circuit considered the \textit{Altai} test, but rejected it for use in this type of case (where expression is not relevant because it is still a method of operation).\textsuperscript{132} Third, the court mentioned the useful article exception for sculptural works but did not apply it in the decision because a computer program is a literary work.\textsuperscript{133} The exception, however, raises the possibility of applying the merger doctrine to deny copyright protection, as many courts have done in connection with the idea-expression analysis. Fourth, the First Circuit noted that certain commands and terms in spreadsheets are "stock" commands that, if covered by copyright, would defeat the principles of copyright law.\textsuperscript{134} Those principles are embodied by the doctrine of scenes \textsc{A} faire such that the Supreme Court could have passed on it along with the merger doctrine and the \textit{Altai} test. And finally, in the concurring opinion, Judge Boudin raised the possibility of privileged use applying to computer software because of the inherent "building" process that spawns better, more efficient programs.\textsuperscript{135} This Part of the Note will address each of the five issues in turn.

A. \textbf{METHOD OF OPERATION TEST}

The only issue on Lotus's appeal to the First Circuit was whether the menu command hierarchy was a copyrightable expression.\textsuperscript{136} As a result, the First Circuit did not have to address a general infringement claim and could focus strictly on the question of whether copyright law protects one particular aspect of the computer program: the menu command hierarchy.

In rejecting the application of the \textit{Altai} test to such a specific aspect of the program,\textsuperscript{137} the First Circuit focused its initial

\textsuperscript{131} \textit{Lotus I}, \textit{supra} note 14, at 818.
\textsuperscript{132} \textit{Id.} at 814-15.
\textsuperscript{133} \textit{Id.} at 817.
\textsuperscript{134} \textit{Id.} at 815.
\textsuperscript{135} \textit{Id.} at 821 (Boudin, J., concurring).
\textsuperscript{136} \textit{Lotus I}, \textit{supra} note 14, at 813. Borland conceded that Lotus had a valid copyright in the spreadsheet program as a whole.
\textsuperscript{137} Because the \textit{Altai} test has been applied differently in many cases, it will hereinafter be referred to as the Abstraction-Filtration-Comparison test, except where there is specific reference to the actual \textit{Altai} application. The following section will further explain the adoption of that test and its proper application to copyright infringement cases involving
inquiry on the issue of protectability. With the proper factual setting, a court may more efficiently address the issue of protectability at the outset rather than rely on the cumbersome abstraction-filtration-comparison analysis. The fact that only one element of the program was at issue presented the First Circuit with such an opportunity but did not justify its decision to disregard completely the Abstraction-Filtration-Comparison process. The ultimate conclusion that the menu command hierarchy is uncopyrightable is correct. However, the analysis leading to that conclusion—that the menu command hierarchy is a method of operation—while true, is not entirely satisfactory. The Supreme Court thus had an opportunity to set forth a more clear-cut framework for a proper initial analysis.

The command hierarchy truly is a functional aspect of the Lotus 1-2-3 program, as the analogy to VCR buttons and labels indicates. As such, copyright law should not protect it from use by other programmers. "[A]ny person may practise [sic] and use the art itself which [the author] has described and illustrated therein." Arguably, the command hierarchy is a guide to the art of spreadsheet programs as the T-accounts illustrated by Selden were a guide to that accounting system. For this reason, the analogy advanced by Borland is applicable, and the First Circuit should have taken it under consideration.

Ruled lines and account headings, as illustrated in Selden's book, are functional because they were made to be used by others.

139 The Court should have performed at least the abstraction-filtration process. Because Borland did copy the hierarchy, no comparison would be needed. Again, this will be discussed more thoroughly in the following section.
141 Borland argued that the only difference between Selden's T-accounts and the menu commands is that the "user interface" of Selden's system was implemented by pen and paper rather than by computer. Lotus I, supra note 14, at 814. Judge Stahl rejected this claim because of the difference in claims: Selden claimed protection of his entire accounting system; Lotus only claimed protection over its "expressive" choice of commands. Id. The fault in this logic is that Selden claimed protection of his actual T-accounts—those illustrated in his book. But because they are simply input functions, as Borland asserted, they were held unprotectable by copyright, even though it was a new expression and organization of material. Similarly, Lotus cannot claim protection of its functions, even though they are arranged in a new expressive fashion.
Generally, illustrations can only be embodied in concrete forms, namely wood, metal, stone, or some other physical form. For example, an illustration of a bird house would be embodied by the actual bird house after being built. The fact that Selden's illustrations can only be embodied by further drawings does not change the fact that they are functional guidelines for accounting, useful beyond their value as a mere illustration. "[T]he principle is the same in all." And the principle is the same if the illustration or diagram is embodied on the screen of a computer.

Even assuming the rejection of the *Baker v. Selden* analogy, the First Circuit could have analogized programming to industrial design to reach the same conclusion. Perhaps such a comparison would create less confusion than the analogy to VCR labels. The only difference between programming and industrial design is that programs (and their features) are based on text rather than physical machinery. The position and placement of a trigger on a mechanical drill could possibly be expressive, but it is a useful method for operating the drill. It performs its function because of the machinery supporting it. Similarly, a command menu, while expressive, is also a useful method for operating the spreadsheet. It performs its function because of the text of the source and object code. Neither the drill trigger nor the command menu is copyrightable, regardless of its expressive qualities.

No matter which analogy it might have chosen, the Supreme Court could easily have concluded that the menu command hierarchy is a method of operation. In addition to the First Circuit, the court in *Mitek Holdings, Inc. v. Arce Engineering Co.* found that the menu structure of the program at issue was a process and thus not copyrightable. In creating a useful method of operation, the author "must seek patronage and protection from its

---

142 *Baker*, 101 U.S. at 105.
143 The Amicus Brief to the Supreme Court stated that one of the major problems with the First Circuit decision was that it did not do a good job of distinguishing method from expression. Amicus Brief, *supra* note 13, at 16.
144 Samuelson et al., *supra* note 72, at 2327-28.
145 This is clearer when considered in connection with the merger doctrine, discussed in part III.C., *infra*.
147 *Id.* at 1580.
utility to the public, and not as a work of science." At least, by affirming the First Circuit opinion, the Supreme Court properly held the menu command hierarchy to be uncopyrightable. What it also did, however, was affirm other problems in the decision by oversimplifying copyright analysis. It placed too much emphasis on the method of operation exception to the exclusion of other important doctrines.

B. PROPER APPLICATION OF THE ALTAI TEST

Courts have readily accepted the use of the Abstraction-Filtration-Comparison test in computer infringement cases. Exactly how to apply it, however, is not so apparent. The circuit courts cannot agree on when to apply the test. Some limit its use to cases concerning nonliteral copying only, while others apply the test even in cases that involve literal copying.

A further problem in the application of the test is that the courts cannot agree on the distinction between literal and nonliteral copying. Judge Stahl declared that the First Circuit was “faced with Borland’s deliberate, literal copying of the Lotus menu command hierarchy.” He conceded, however, that Borland did not copy any of the Lotus 1-2-3 underlying code. The problem with that analysis is that literal copying usually refers to copying

---

148 Baker v. Selden, 101 U.S. 99, 105 (1879). This means that patent law, if any, is the only appropriate law for protecting even an expressive method of operation.
149 See supra note 65 (giving examples of the warm embrace).
150 "Court decisions are, generously described, in a state of creative ferment concerning the methods by which nonliteral elements of computer programs may be identified and analyzed for copyrightability." Engineering Dynamics, Inc. v. Structural Software, Inc., 26 F.3d 1335, 1341 (5th Cir. 1994).
151 Bateman v. Mnemonics, Inc., 79 F.3d 1532, 1545, 38 U.S.P.Q.2d (BNA) 1225 (11th Cir. 1996). That court cites Lotus I to illustrate an example of a court that refused to apply the analysis to a situation that it considered literal copying. Id. The district court in the Bateman case also limited the test to cases involving nonliteral copying. Id. at 1544 (providing excerpt of district court's opinion). That limitation was one of the primary reasons the case went up to the Eleventh Circuit on appeal. Id. at 1543-44. The district court, in Mitek v. Arce also confronted the problem and acknowledged the confusion: it looked back to Gates (which expanded on the Altai test) and determined that Gates only applies to literal copying. Mitek Holdings, Inc. v. Arce Eng'g Co., 864 F. Supp. 1568, 1578 (S.D. Fla. 1994).
152 Lotus I, supra note 14, at 814 (emphasis added).
153 Id. at 810.
the code. For example, the Bateman court questioned application of the Abstraction-Filtration-Comparison test because that case involved literal copying: Mnemonics had admitted to copying portions of the code.\textsuperscript{154} Similarly, the Mitek court found the test applicable to "literal aspects of the program, i.e. the source code and object code because it does not manifest itself visually."\textsuperscript{155}

The confusion over when to apply the Altai test clearly demonstrates the need for a uniform rule to guide the circuit courts in their decision making. However, the distinction between literal and nonliteral copying is really "more of a matter of semantics than substance."\textsuperscript{156} The Bateman court asserted that the formal test would certainly apply to nonliteral copying (elements other than the underlying code), but recognized the need for a similar analysis when analyzing the code. Still, the court refused to offer an opinion as to whether the exact same test or evaluation should apply.\textsuperscript{157}

Because the same question was at issue in Lotus I, where the First Circuit rejected both the standard Abstraction-Filtration-Comparison test and any variation for its method-expression distinction, the Supreme Court had an opportunity to address the question on appeal.

1. Why The Abstraction-Filtration-Comparison Test Should Have Applied. Although a computer program as a whole falls under the category of "literary work,"\textsuperscript{158} the program can be divided into different elements. Upon division, the screen display is entitled to protection as a separate literary work, a pictorial or graphic work, or an audio-visual work.\textsuperscript{159} The screen display itself has several independent elements, one of which is a menu.\textsuperscript{160} Because so

\textsuperscript{154} Bateman, 79 F.3d at 1542 (emphasis added).

\textsuperscript{155} Mitek, 864 F. Supp. at 1581.

\textsuperscript{156} Bateman v. Mnemonics, Inc., 79 F.3d 1532, 1545 (11th Cir. 1996).

\textsuperscript{157} Id.


\textsuperscript{159} Peter S. Menell, An Analysis of the Scope of Copyright Protection for Application Programs, 41 STAN. L. REV. 1045, 1084 (1989).

\textsuperscript{160} "A ‘menu,’ in computer parlance, is a graphical user interface employed to store information or functions of the computers in a place that is convenient to reach, but saves screen space for other images." Mitek Holdings v. Arce Eng’g Co., 864 F. Supp. 1568 n.11 (S.D. Fla. 1994) (citing Apple Computer Inc. v. Microsoft Corp., 799 F. Supp. 1006, 1037 (N.D. Cal. 1992)). The Ninth Circuit in Apple v. Microsoft found that the menu as a whole is an idea not covered by copyright, but did not assess the menu command hierarchy, which is a part of the menu. Apple Computer, Inc. v. Microsoft Corp., 35 F.3d 1435, 1444 (9th Cir.
many elements are present on a computer display and can be arranged in so many different ways, the entire screen or even elements within the screen could receive copyright protection as a compilation.\textsuperscript{161} Like a menu, the menu command hierarchy has individual elements: the commands themselves. In fact, Lotus 1-2-3 has 469 commands arranged into more than 50 menus and submenus.\textsuperscript{162} Unlike the First Circuit in \textit{Lotus I}, however, the Ninth Circuit acknowledged that a claim could rest on the unique selection and arrangement of all the features in the interface.\textsuperscript{163} Similarly, the Fifth Circuit found that a series of formats, when taken together as a whole, qualify as an expressive arrangement where they "do not self-evidently convey only an 'idea.'"\textsuperscript{164}

The Tenth Circuit refused to address the question of protectability of menus because the record from the district court was too ambiguous in explaining what a menu was and how it might receive protection.\textsuperscript{165} Because different interpretations are possible, and because the circuits appear to remain uncertain about protecting menus, the Supreme Court could have clarified the issue in \textit{Lotus II}. The First Circuit ruled that the menu command hierarchy as a whole was uncopyrightable, yet focused its analysis (during the VCR analogy) on the individual commands. It thus

\textsuperscript{161} Brown, supra note 29, at 1003. Brown argues that most displays of programs used by businesses would satisfy the requirements of a compilation under § 101 of the Copyright Act, including the menu of Lotus 1-2-3. \textit{Id.} "A 'compilation' is a work formed by the collection and assembling of preexisting materials or of data that are selected, coordinated, or arranged in such a way that the resulting work as a whole constitutes an original work of authorship." 17 U.S.C. § 101. Brown asserts that while this is the correct application of the law, it is one of the many problems encountered when applying copyright law to computer programs. Brown, supra note 29, at 1003.

\textsuperscript{162} \textit{Lotus I}, supra note 14, at 809.

\textsuperscript{163} \textit{Apple Computer, Inc. v. Microsoft Corp.}, 35 F.3d 1435, 1446 (9th Cir. 1994), \textit{cert. denied}, 115 S. Ct. 1176 (1995). As just explained, the menu command hierarchy can be broken down into its own sub-elements just like an interface can.

\textsuperscript{164} \textit{Engineering Dynamics, Inc. v. Structural Software, Inc.}, 26 F.3d 1335, 1344 (5th Cir. 1994). Formats are a part of the program's code, but a similar analysis can apply. While the court in that case did not address the issue of compilations, the result is the same—a combination of elements that may be individually unprotected for whatever reason may nonetheless receive protection as a whole because of their expressive arrangement.

\textsuperscript{165} \textit{Gates Rubber Co. v. Bando Chem. Indus.}, 9 F.3d 823, 843 (10th Cir. 1993). In defining "menu," the 10th Circuit cited as an example the definition from the district court in \textit{Lotus v. Borland}. 
failed to acknowledge that even if the commands are unprotectable, there is still a possibility that the arrangement of the commands and their structure could receive protection.\(^{166}\) For that reason, the Supreme Court should have analyzed the case through some sort of Abstraction-Filtration-Comparison test to determine if the menu command hierarchy, apart from its individual commands, deserved protection, or at least should have set some guidelines for a proper determination on remand.

2. Application Of An Abstraction-Filtration-Comparison Test. Even though Lotus has a valid copyright on Lotus 1-2-3, and even if the menu command hierarchy is distinguishable from its individual commands as a compilation or other “expressive” work, an application of the Abstraction-Filtration-Comparison test should still render the menu command hierarchy uncopyrightable.

A work that is subject to copyright protection when considered as a whole may still contain elements that cannot be protected individually.\(^{167}\) Even though an element of a program may be thought of as the “look and feel” of a program, and thus considered expressive, it can only receive protection if another doctrine does not remove it from the coverage of copyright law.\(^{168}\) Because even these expressive elements may be unprotectable, they must be dissected.\(^{169}\) The arrangement of commands within the Lotus 1-2-3 menu command hierarchy may be expressive, but because it may also be within another category excluded from copyright, an Abstraction-Filtration-Comparison test was certainly appropriate for use by the Supreme Court. The only question should have been how to apply it.

\(^{166}\) As stated in Gates, “unprotectable elements of a program, even if copied verbatim, cannot serve as the basis for ultimate liability for copyright infringement. However, the copying of even unprotected elements can have a probative value in determining whether the defendant copied the plaintiff's work.” 9 F.3d at 832 n.7. The same principle would apply even if analyzing one element of the program, such as a menu or menu command hierarchy.

\(^{167}\) Feist Publications, Inc. v. Rural Tel. Serv. Co., 499 U.S. 340, 348 (1991) (“The mere fact that a work is copyrighted does not mean that every element of the work may be protected”).

\(^{168}\) Apple Computer, Inc. v. Microsoft Corp., 35 F.3d 1435, 1439 (9th Cir. 1994), cert. denied, 115 S. Ct. 1176 (1995). Some of the possible doctrines that would remove the element from protection are any of the specific characterizations found under 17 U.S.C. § 102(b), merger, scenes à faire, fair use, etc.

\(^{169}\) Id. This “dissection” refers to the Abstraction-Filtration-Comparison test.
a. Abstraction. As noted above, the Altai test was the first application of the Abstraction-Filtration-Comparison process. The Tenth Circuit accepted the basic analysis but expanded on it to set forth affirmative guidelines in its application. That court also recommended comparing the programs in their entirety before beginning the dissection. The advisability of such a comparison, however, is a subject of considerable debate. Once again, the Supreme Court had the opportunity to settle a dispute, but failed to do so.

What is not in dispute is the difficulty of analyzing an individual claim. When breaking down a work into a series of abstractions, a court will reach a point somewhere in the series where the abstraction is not protectable. However, "[n]obody has ever been able to fix that boundary, and nobody ever can." Although the line may seem somewhat arbitrary, a court must face its responsibility and draw the distinctions. Indeed, many courts have managed to draw the line quite satisfactorily. Courts should take a case-by-case approach in their abstraction analy-

---

170 See supra note 64 and accompanying text.
171 Gates Rubber Co. v. Bando Chem. Indus., 9 F.3d 823, 841 (10th Cir. 1993). The court specified 6 different levels of generality: (1) the main purpose, (2) the program structure, (3) modules, (4) algorithms, (5) source code, and (6) object code. Id. at 835. This list is by no means exhaustive. Of those levels listed, the main purpose is almost always not protectable. On the other hand, the source code and object code usually are protectable. Id. at 836. These specific levels of abstraction apply to the underlying, "literal" elements of the software. The same principles apply to nonliteral elements, but obviously these specific abstractions do not.
172 Id.
173 See, e.g., Apple Computer, Inc. v. Microsoft Corp., 35 F.3d at 1443 (must determine questions of what and how much can be protected before considering work as a whole).
174 Nichols v. Universal Pictures Corp., 45 F.2d 119, 121 (2d Cir. 1930). The reason for reaching such a point is that if all were protectable, the author of the work would be able to prevent the use of his ideas, not just the expression of them. That would clearly run against the very nature of copyright protection.
175 Id. (emphasis added). This famous statement by Judge Learned Hand takes on a new meaning in its application to computer software. The abstractions from software are much more complex than normal literary or artistic expression.
176 Id. at 122.
177 Amicus Brief, supra note 13, at 24 (noting the success of the Altai, Gates, and Apple v. Microsoft courts in finding elements unprotectable because of their functionality).
178 Engineering Dynamics, Inc. v. Structural Software, Inc., 26 F.3d 1335, 1343 (5th Cir. 1994) ("Protectable originality can manifest itself in many ways, so the analytic approach may need to be varied to accommodate each case's facts"); Gates Rubber Co. v. Bando
ses because so much depends on the individual programs, whether or not copying is admitted, and whether that copying is literal or nonliteral. The difficulty in the abstraction process should not have steered the First Circuit away from addressing the issue by choosing its own method of analysis. Taking notice of the successful efforts of other circuits, the Supreme Court at least should have required the lower courts in the *Lotus* case to rise to the challenge by attempting their own abstraction analysis.

b. Filtration. The filtration process should eliminate ideas, processes, facts, public domain information, merger material, scenes à faire material, and any other element that the Copyright Act specifically excludes from the scope of its protection. The authors of the Amicus Brief for *Lotus II* argued correctly that the Court could clearly filter processes and methods from the program along with general and abstract ideas. Thus the First Circuit was not wrong in concluding that the Lotus 1-2-3 menu command hierarchy could not be protected as a method of operation—it merely proceeded under a faulty analysis. The Supreme Court should have confirmed filtration, and not the analysis adopted by the First Circuit, as the appropriate test for eliminating uncopyrightable material from the scope of copyright protection.

The filtration step applies with equal force to computer screen displays as it does to useful code language. Determining what should be eliminated during the filtration step is, however, subject

---


179 Gates, 9 F.3d at 834.

180 Amicus Brief, supra note 13, at 14.

181 See generally Mitek Holdings, Inc. v. Arce Eng’g Co., 864 F. Supp. 1568, 1579-84 (S.D. Fla. 1994) (filtering out 13 of 18 screen elements as unprotectable for various reasons). As discussed earlier, supra notes 149-156 and accompanying text, the entire Abstraction-Filtration-Comparison test applies to nonliteral elements, but not simply by examining the code underlying those elements. As the *Lotus I* court attempted to explain, the elements on the screen actually cause the operation of the desired program function by selecting the proper command. This is particularly so when dealing with a Windows-based program because then a direct “double-click” on the relevant icon actually activates the program. Thus, a screen feature may be clearly functional and not subject to copyright protection when analyzed under this test.
to each court's own interpretation. In *Apple v. Microsoft*, the court filtered out most of the expressed features because Microsoft had licensed them from Apple's graphical interface. In an older case before the time of computers, the Supreme Court ruled that the basic symbols that make up a shorthand system were not copyrightable because they were an inherent part of the shorthand language. The Amicus Brief to the Supreme Court in *Lotus v. Borland* argued that the same principle should apply to a set of computer commands. In other words, because commands are an inherent part of the macro language used to operate the computer program, those commands and their organization are not copyrightable. In *Ashton-Tate v. Ross*, the Ninth Circuit denied protection to a computer software command structure because that structure was too abstract to be protectable by copyright.

These three cases suggest that even if the Lotus 1-2-3 menu command hierarchy is a compilation separate from the functionality of the underlying commands, a court would still be justified in denying copyright protection. Under the principle of *Apple v.*
Microsoft, the Court would have to proceed to the comparison stage, but Brief English Systems and Ashton-Tate suggest that the Court could immediately eliminate the entire menu command hierarchy from consideration. At the very least, the Court could eliminate the menu commands themselves before reaching the comparison stage because the commands clearly are not entitled to protection standing alone.

On the other hand, the Engineering Dynamics court classified the Lotus 1-2-3 menu command hierarchy as purely expressive. That court, like the Lotus I court, assumed that the command hierarchy could be examined as a whole for purely expressive characteristics. Thus, it also ignored the fact that the menu command hierarchy can be broken down into further levels of abstraction. After filtering those lower levels of abstraction, the examination would be different. Only after that further filtration can the jury or court continue to the comparison stage.

c. Comparison. In this stage of the analysis, Gates Rubber explicitly rejects the “total concept and feel” comparison. Such a comparison is of little value after the filtration stage. In the case of Lotus 1-2-3 and Quattro Pro, the commands would no longer be a part of the analysis at this point in the test. Because virtually every element of the menu command hierarchy is uncopyrightable, the Court should have also found the structure as a whole uncopyrightable.

188 Following in the next subsection.
189 Both of these cases suggest an analysis akin to applying the merger doctrine, discussed in part III.C., infra.
190 As discussed earlier, the individual commands are functional, abstract, inherent parts of a language, etc., all of which are not protected by copyright. They are also stock phrases, as will be discussed with respect to merger and scenes à faire in the next two sections, III.C. and III.D.
191 Engineering Dynamics, Inc. v. Structural Software, Inc., 26 F.3d 1335, 1344 (5th Cir. 1994). That court agreed with Judge Keeton’s findings from the district court level and stated that the utilitarian nature of the commands does not change the expressive nature of the structure as a whole. Id. at 1346.
192 Gates Rubber Co. v. Bando Chem. Indus., 9 F.3d 823, 839 n.15 (10th Cir. 1993). That is why the court suggested an examination as to the total concept and feel prior to dissection. After the unprotectable elements are gone, very little is left to enable the court to get a “feel” for the total concept. Presumably, those elements providing the concept are functional and will have been eliminated.
193 Although this seems contrary to the purpose of “compilations” under the Copyright Act, the anomalous result arises because of the dual nature of software—utilitarian and expressive. The commands themselves are inherently useful, thus any efficient arrangement will also be useful, even if also expressive. Such a tedious Abstraction-Filtration-Comparison
One additional problem arises in the comparison stage that would possibly allow Lotus to win its infringement claim: many courts allow “thin” protection even where virtually every underlying element or abstraction is unprotectable. The Engineering Dynamics court asserted that the same careful approach established and adopted by Feist should also apply to cases involving computer interfaces. In such cases, verbatim copying would still constitute infringement. Other courts apply an even more lenient standard in thin protection cases: virtually identical copying. If the thin protection were to apply, Borland would be liable for infringement. It not only virtually copied the Lotus 1-2-3 command hierarchy, it copied the feature verbatim, thus satisfying even the stricter standard.

Not all courts, however, apply the thin protection standard. As the Ninth Circuit stated, “[i]n some circumstances, even the exact set of commands used by a programmer is deemed functional rather than creative for the purposes of copyright” and thus would not be

---

194 Thin protection is protection only against verbatim copying. In traditional cases, “verbatim” has an obvious meaning. In software cases, the troubling distinction between literal and nonliteral copying arises with respect to whether verbatim copying applies to the actual code (from the diakettes) or to verbatim reproduction of the visual effects, etc.

195 Engineering Dynamics, Inc. v. Structural Software, Inc., 26 F.3d 1335, 1348 (5th Cir. 1994) (citing district court in Lotus v. Borland, 831 F. Supp. 202, 209 (D. Mass. 1993)). This allowance for infringement claims based on verbatim copying is particularly useful for plaintiffs trying to prevent reverse engineering of their programs. In reverse engineering cases, the new programmer can legally search an existing program for unprotectable ideas and functions to use, but the process requires the making of an identical copy to break down into its component parts. Plaintiffs often base their claims on the actual copying because it is too hard for them to prove protection after the filtration process. David A. Rice, Sega and Beyond: A Beacon for Fair Use Analysis . . . At Least as Far as It Goes, 19 U. DAYTON L. REV. 1131, 1136 (1994). See part III.E., infra, on reverse engineering and its relation to fair use.

196 Id.


198 Only the commands and their on-screen structure were copied, not the underlying code.
The Supreme Court would have been unwise to allow even thin protection for such a functional aspect of the Lotus 1-2-3 program. Thin protection still limits access to utilitarian expression, thus preventing other programmers from using even the most basic expression to design a better product. Because the First Circuit did not allow such protection, the Supreme Court, by its simple affirmation, has not allowed it either. Unfortunately, neither decision expressly rejects thin protection, and thus it remains a viable basis for copyright protection.

C. MERGER DOCTRINE

The First Circuit, during its analogy to the buttons and labels of a VCR, raised the point that if the buttons of a VCR had been purely expressive in their arrangement, they might still have been subject to a useful-article exception. Judge Stahl, however, distinguished the commands from buttons because computer software, unlike VCRs, is defined by the Copyright Act as literary work. Although the useful-article exception may itself not apply, the closely related doctrine of merger should.

199 Sega Enter. Ltd. v. Accolade, Inc., 977 F.2d 1510, 1524, 24 U.S.P.Q.2d (BNA) 1561 (9th Cir. 1992). That court analyzed the code rather than the features of a screen display, but the same principles apply. The code is generally considered the most expressive aspect of a program, so if it cannot receive some copyright protection because it is functional, neither should anything else.

200 Rice, supra note 195, at 1140. The policy rationale behind disallowing protection for functional elements will be discussed along with merger in the next section.

201 "A 'useful article' is an article having an intrinsic utilitarian function that is not merely to portray the appearance of the article or to convey information." 17 U.S.C. § 101 (1988 & Supp. 1993).

202 Lotus I, supra note 14, at 817. The useful-article exception only applies to a pictorial, graphic, or sculptural work. 17 U.S.C. § 101. One commentator has argued the contrary, that a program could be considered a useful article. Phillips, supra note 9, at 1017 (citing Lotus Dev. Corp. v. Paperback Software Int'l, 740 F. Supp. 37 (D. Mass. 1990). He explained that the Paperback court did not apply the exception, however, because Judge Keeton felt that copyright law had advanced under congressional and judicial developments to preclude such an application. Id. See also Menell, supra note 158, at 1049-50 (explaining why literary classification is not sound because of overly broad protection without consideration of proper application of merger doctrine).

203 Judge Stahl even noted the concept of merger in relation to the protection of Lotus's "long prompts"—another feature that was not on appeal. His footnote suggests that the First Circuit would recognize the merger concept as a defense to infringement: "when merger occurs, identical copying is permitted." Lotus I, supra note 14, at 816 n.9. But see Apple
In order for copyright protection to attach to a work, the expression must be distinct from the idea. "Copyright protection is denied to expression that is inseparable from or merged with the ideas, processes, or discoveries underlying the expression." This theory acts as a "prophylactic device" to protect ideas. The dissenting commissioner in the CONTU Final Report obviously had this principle in mind. He did not like the notion of classifying computer software as a literary work because programs are similar to other objects "designated to do work—for example, the cam of a drill." Allowing such expression to be protectable would insulate the ideas from effective use as well.

Some academics argue that merger is not applicable in a case such as *Lotus* or other software cases because merger only applies where the particular expression is "essential" to the underlying idea, process or function. Just because some expression is the best or most efficient does not make it essential for the purpose of applying a merger analysis; merger applies only where no different expression is conceivable. However, other academics and practitioners would not necessarily agree with such a strict application of merger. Those critics would argue that "the availability of alternative choices is not by itself a reliable basis for distinguishing between elements of a program that are expressive and those that are excludable under § 102(b)."

Computer, Inc. v. Microsoft Corp., 35 F.3d 1435, 1444 (9th Cir. 1994), cert. denied, 115 S. Ct. 1176 (1995) (recognizing thin protection even if idea and expression are merged; would prohibit nearly identical copying). Gates Rubber Co. v. Bando Chem. Indus., 9 F.3d 823, 838 (10th Cir. 1993). That court applied the merger doctrine in the filtration process of its Abstraction-Filtration-Comparison test. The Lotus 1-2-3 menu command hierarchy's expressive structure could merge with an idea or process as a whole and eliminate the need for a lengthy formal analysis. Or, as advocated earlier, the merger doctrine could simply apply when dissecting the command hierarchy as a compilation in itself.

CONTU Final Report, supra note 35, at 29-30 (Hersey, Comm'r dissenting).

Phillips, supra note 9, at 1020 (arguing that merger doctrine is inadequate to protect ideas in software cases; rather Congress should create *sui generis* protection with variation of merger doctrine as one fundamental aspect). But see Menell, supra note 158, at 1100 (advocating new approach to merger doctrine for more closely tailored protection of software ideas and functions).

This suggests that perhaps scenes à faire would be a better approach.

Amicus Brief, supra note 13, at 37.

Id.
Choices of terms, both in the code and on the screen, more easily merge with their ideas because of the very nature of software. Programs compete on the basis of behavior. Whether the source code or user interface is particularly original or expressive is irrelevant to the competition: no one truly buys a program for that. People buy programs and compare them to other programs based almost solely on how well they operate and how easy they are to operate. 211

Still, courts may leave the question of merger to a jury and allow copyright protection to some interface specifications even where the expressive choices are limited. The jury must simply be informed as to the constraints on choice and the possible application of merger principles. 212 Thus, even where the fact-finder determines that merger does not prevent infringement, it must still consider the doctrine. 213 Menell argues that although copyright law is not well-suited for protection of computer programs, it can be adapted for better application. 214 A careful application of the merger doctrine will help copyright law adapt to software needs because it will stimulate invention and development of better programs. 215 Had the Supreme Court rendered an affirmative opinion in *Lotus II*, it certainly could have addressed the issue of merger and set forth some definitive standards to help tailor the doctrine to the needs of computer programs. Merger is a judicially-created

---

211 Samuelson et al., *supra* note 72, at 2316.
212 Bateman v. Mnemonics, Inc., 79 F.3d 1532, 1545 (11th Cir. 1996).
213 Gates Rubber Co. v. Bando Chem. Indus., 9 F.3d 823, 836, 842 (10th Cir. 1993) (finding doctrine appropriate and holding district court in error for refusing to consider merger in Abstraction-Filtration-Comparison analysis); Mitek Holdings, Inc. v. Arce Eng'g Co., 864 F. Supp. 1568, 1582 (S.D. Fla. 1994) (finding that merger applies because of limited ways to depict three-dimensional plane in design programs).
214 Menell, *supra* note 159, at 1050 n.20.
215 *Id.* at 1050. This assertion is based on the assumption that merger will allow for standardization of certain aspects and elements of software. As an expression becomes the industry standard, it will then merge with the idea so that even identical copying is always permitted. But see Phillips, *supra* note 9, at 1019 (arguing that such application of merger principles is erroneous because expression becomes protectable when *first* expressed, with protection lasting for 75 years; only after it has been protected does it later become industry standard). Menell would respond by advocating a principle similar to trademark law: give protection until the new interface becomes the industry standard. That way the originator has an incentive to create because he will have a market lead until use becomes common. Menell, *supra* note 159, at 1101-02.
doctrine,\textsuperscript{216} thus the Court could have clarified how it should be applied to computer cases. In doing so, it could have, and perhaps should have, followed the guidance advocated by Menell: allow protection until there are no realistic options left for alternative expression, then begin to apply the merger principles.

D. SCENES À FAIRE DOCTRINE

A court may “deny protection to those expressions that are standard, stock, or common to a particular theme or setting.”\textsuperscript{217} This principle is similar to the doctrine of merger and derives from the \textit{Feist} decision: if an expression is usual, it lacks the originality necessary for protection.\textsuperscript{218} Because most courts recognize this as an inherent limitation on copyrightability,\textsuperscript{219} the Supreme Court could have addressed this in \textit{Lotus II} as well, particularly because the First Circuit had noted the stock terms in the menu command hierarchy such as “copy” and “print.”\textsuperscript{220}

Scenes à faire is particularly applicable in computer cases because of the constant drive toward standardization among application programs.\textsuperscript{221} Aside from the desire to make programs more efficient, programmers share a broader goal to make programs more user-friendly—easier to operate and easier to switch from one program to another if necessary.\textsuperscript{222} This is because “[a]

\textsuperscript{216} There is no statutory definition; it is simply a part of copyright’s common law. The doctrine arises out of the idea-expression dichotomy and is essentially an extension of the rationale announced in \textit{Baker v. Selden}. \textit{See} Morrissey v. Procter & Gamble Co., 379 F.2d 675, 154 U.S.P.Q. (BNA) 193 (1st Cir. 1967) (expanding on rationale of \textit{Baker v. Selden} to introduce what is now known as merger doctrine). “When the uncopyrightable subject matter is very narrow, so that ‘the topic necessarily requires,’ . . . if not only one form of expression, at best only a limited number, to permit copyrighting would mean that a party or parties, by copyrighting a mere handful of forms, could exhaust all possibilities of future use of the substance.” \textit{Morrissey}, 379 F.2d at 678 (citation omitted).

\textsuperscript{217} Gates Rubber Co. v. Bando Chem. Indus., 9 F.3d 823, 838 (10th Cir. 1993).


\textsuperscript{219} \textit{See}, e.g., \textit{Gates}, 9 F.3d 823 (recognizing such inherent limitation).

\textsuperscript{220} \textit{Lotus I}, \textit{supra} note 14, at 815.

\textsuperscript{221} \textit{See generally} BEN SHNEIDERMAN, \textbf{DESIGNING THE USER INTERFACE: STRATEGIES FOR EFFECTIVE COMPUTER INTERACTION} (1987) (explaining goals and attempts of programmers to achieve some sort of uniformity in computer software).

\textsuperscript{222} Menell, \textit{supra} note 159, at 1052.
Program that is not easily used is a program that will not be used."²²³ Programmers no longer want to create a unique, customized design or application. Instead, they use a more methodological, engineering approach.²²⁴ This drive toward uniformity inevitably causes certain specifications that might have been originally expressive to become stock or standardized.

CONTU overlooked the public desire for standardization when it prepared its study.²²⁵ According to a national trade study, 99.3% considered ease of use as a primary factor in considering what software to buy.²²⁶ Another 86.4% factored compatibility into the equation.²²⁷ Unlike CONTU, however, several courts have already taken standardization into consideration.²²⁸ The Court in Lotus II should also have considered the commands in the menu command hierarchy, and perhaps the entire structure as a whole, to be standards in the industry. Most spreadsheet programs have copied some aspects of Lotus 1-2-3 because it was "the spreadsheet leader, the state of the art."²²⁹ The Court could clearly have found the individual commands of Lotus 1-2-3 to be stock phrases and uncopyrightable for purposes of the Abstraction-Filtration-Comparison test. Otherwise it should have remanded the case for a determination of fact as to whether the command hierarchy has

²²³ Ramos, supra note 40, at 277. In order to make programs easier to use, the programmers strive to achieve five goals in particular: (1) minimize learning time, (2) maximize speed of performance, (3) minimize rate of user errors, (4) maximize user satisfaction, (5) maximize users' retention of knowledge over time. SHNEIDERMAN, supra note 221, at 14-15.

²²⁴ Samuelson et al., supra note 72, at 2331.

²²⁵ Menell, supra note 159, at 1066.


²²⁸ See, e.g., Gates Rubber Co. v. Bando Chem. Indus., 9 F.3d 823, 838 (10th Cir. 1993) (denying protection where a certain expression is "dictated by external factors," namely specific software standards and target industry demands); Engineering Dynamics, Inc. v. Structural Software, Inc., 26 F.3d 1335, 1347 (5th Cir. 1994) (allowing district court to consider industry demands and practice on remand); Mitek Holdings, Inc. v. Arce Eng'g Co., 864 F. Supp. 1568, 1582 (S.D. Fla. 1994) (finding word "cut" to be term of art and unprotectable under scenes à faire doctrine).

²²⁹ Brown, supra note 29, at 1004 (emphasis added).
become an industry standard, with instructions to find for Borland if it has.\textsuperscript{230}

E. PRIVILEGE

Judge Boudin, in his concurring opinion in \textit{Lotus I}, suggested that while the position of Judge Stahl was justified, he might prefer a different analysis. Instead of determining that the menu command hierarchy is completely unprotectable, Boudin would allow Borland (and others similarly situated) a privileged use.\textsuperscript{231} This mode of analysis would allow Lotus to reap the rewards of its creation by prohibiting other programmers from simply copying nonliteral aspects and reselling them under their own labels.\textsuperscript{232} Yet it would allow companies such as Borland to reap the rewards of making a better product whose only use for the Lotus menu structure was to enable the users to build on their own investment in learning.\textsuperscript{233} As Judge Boudin explained, "[T]he closest analogue [to this allowance] in conventional copyright is the fair use doctrine."\textsuperscript{234} Fair use only applies when the material is copyrightable from the outset, not when protection is denied. Currently, only cases of reverse engineering have applied the fair use doctrine to computer programs.\textsuperscript{235} Copying is essential to reverse engineering techniques, and reverse engineering is desirable because it allows programmers to get to the core functionality of a program. Without it, the holder of a copyright would be able to protect ideas and

\textsuperscript{230} The same problems discussed in the merger section with respect to initial copyrightability and a subsequent loss in protection would also be applicable to the scenes à faire doctrine. This Note suggests adopting an analysis similar to that asserted by Menell, \textit{supra} note 159, at 1101. Balancing the incentive to create with dissemination of ideas would best accomplish the goals of copyright.

\textsuperscript{231} \textit{Lotus I, supra} note 14, at 821 (Boudin, J., concurring).

\textsuperscript{232} \textit{Id.}

\textsuperscript{233} \textit{Id.} Boudin explains that users would not choose Borland for the Lotus menu—otherwise they would choose the program that created the menu, for whom the menu was designed. Borland's use is simply a fallback option; its success is really due to its other improvements and new features. \textit{Id.} at 820.

\textsuperscript{234} \textit{Id.} at 821.

\textsuperscript{235} \textit{Atari Games Corp. v. Nintendo of Am., Inc.,} 975 F.2d 832, 843, 24 U.S.P.Q.2d (BNA) 1015 (Fed. Cir. 1992) (holding it fair use for programmer to reverse engineer program, including making of exact copy, to find unprotectable ideas for his own use).
functions that are not otherwise protectable. 236 Only one other case, Bateman v. Mnemonics, considers fair use in terms of copying nonliteral elements from a graphical interface. 237 There, the Eleventh Circuit held fair use to be a factual issue. 238 Section 107 of the Copyright Act sets out the four factors of a fair use analysis, 239 but because few courts have applied the doctrine to computer programs, a fact-intensive application would be very difficult.

Boudin’s suggestion has definite appeal, but even he admits that it would have its own costs. 240 Certainly the Supreme Court could have laid some groundwork for such a proposition, but other, more solid foundations already exist which the Court could have simply expanded or clarified. The Abstraction-Filtration-Comparison, along with the merger and scenes à faire doctrines, would have been more appropriate areas for the Court’s exploration. As it stands now, however, courts must continue to hash out their own rulings based on the facts of each individual case, even if that leads to inconsistent results.

IV. IMPLICATIONS OF NON-DECISION

Because the Supreme Court offered no guidance in its opinion and the First Circuit decision appears confined to a limited set of circumstances, other circuits have nowhere to turn but to copyright policy. 241 CONTU drew the line between ideas and expression with the knowledge that drawing it too liberally would grant monopolies inhibiting improvement while drawing it too conservatively would discourage efforts to create. 242 Protecting ideas that

236 Rice, supra note 195, at 1136.
237 79 F.3d 1532 (11th Cir. 1996).
238 Id. at 1547 n.32 ("And even if the expression is found to be original, fair use may be demonstrated by the alleged infringer to defeat the claim of infringement").
240 Such costs would include administrative difficulties and lack of predictability. Lotus I, supra note 14, at 821-22 (Boudin, J., concurring).
241 Of course prior cases and precedent will still control, but copyright policy provides the best analytical framework for the courts to decide a current case—each analysis is on a case-by-case basis. Engineering Dynamics, Inc. v. Structural Software, Inc., 26 F.3d 1335, 1343 (5th Cir. 1994); Gates Rubber Co. v. Bando Chem. Indus., 9 F.3d 823, 834 (10th Cir. 1993).
underlie even the most expressive programs and their elements would make costs of standardization so high that no one could build on another's ideas.\textsuperscript{243} Such protection would run directly contrary to the constitutional mandate that copyright be used to "promote the Progress of Science and useful Arts."\textsuperscript{244}

As the First Circuit noted in \textit{Lotus I}, building on others' ideas is central to both copyright and computer programming.\textsuperscript{245} Programs are built from other smaller programs; lots of sub-programs combine to make one complex program.\textsuperscript{246} Moreover, programming is a give and take process.\textsuperscript{247} Programmers commonly adopt each other's design elements and ideas for solving various problems in different contexts. This is evident in the \textit{Lotus} case. The idea of a spreadsheet did not originate with Lotus, but with a creative-but-frustrated MBA student who devised a program called VisiCalc for help in his personal studies. Lotus was just one of several spreadsheet programs that followed VisiCalc's lead.\textsuperscript{248} After that second generation of software, improvements came quickly. Quattro Pro is a superior program with faster operation, better graphics, and more capabilities than its predecessors, such as Lotus 1-2-3.\textsuperscript{249}

All of these advances result from a building process. Programming is based on skilled know-how—like industrial design—everything derives from an existing idea.\textsuperscript{250} Although rewarding the "authors" of these programs is one implicit goal of copyright law, it is not as vital as encouraging advancement for the betterment of society and the general public.\textsuperscript{251}

\textsuperscript{243} Gates Rubber Co. v. Bando Chem. Indus., 9 F.3d 823, 838 (10th Cir. 1993).
\textsuperscript{244} U.S. CONST. art. I, § 8, cl. 8.
\textsuperscript{245} \textit{Lotus I}, supra note 14, at 818.
\textsuperscript{246} Samuelson et al., supra note 72, at 2326.
\textsuperscript{247} Id. at 2330-31.
\textsuperscript{248} See Menell, supra note 159, at 1057 (describing creation of spreadsheet software).
\textsuperscript{249} Mike Hogan, \textit{Product Outlook: Fresh from the Spreadsheet Oven}, PC WORLD, Feb. 1988, at 100-102 (describing improvements in spreadsheet software).
\textsuperscript{250} Samuelson et al., supra note 72, at 2331. This shows how patent protection appears to be more appropriate for software. However, many commentators do not like patent law any more than copyright, especially with the direction copyright has begun to move in post-\textit{Altai} times. Others, including Samuelson, advocate a \textit{sui generis} regime to fit the unique requirements of the software industry.
While patent or a sui generis system of protection might solve some problems associated with the application of copyright law, the "use of copyright law has proceeded so far that the basic policy choice is now irreversible."\textsuperscript{252} Furthermore, despite declaring programs to be statutory material, Congress left the adaptation process to the courts.\textsuperscript{253} Thus the courts must strike the balance between too much and too little protection. The Supreme Court failed in its duty to contribute its steadying hand to that balancing process.

Too much legal protection encourages programmers to use non-compatible methods of expression rather than to improve existing programs and results in an inefficient use of resources.\textsuperscript{254} Too little protection may leave all programmers using the same expression even if better methods might be available, thus causing software advancement to slow and potentially remain in an inferior state for some time.\textsuperscript{255} Recently, universities funded by government and private industry have helped develop newer programs and programming techniques.\textsuperscript{256} In doing so, these funds alleviate some of the problems with stagnation that arise when less protection is available.

The Supreme Court should have set a standard whereby broad copyright protection is rare in computer programs. As the number of patents granted for software increases, a creator should rely more heavily on patents for processes and reserve the use of copyright solely for expression where available. Otherwise, the easy access to and long duration of copyright would provide

\textsuperscript{252} Rice, \textit{supra} note 195, at 1131.
\textsuperscript{253} \textit{Id.} at 1132.
\textsuperscript{254} Menell, \textit{supra} note 159, at 1069. Obviously waste cuts against promoting the advancement of computer software. Judge Keeton, however, focused more on the protective aspect of copyright law rather than societal/constitutional policies when he observed that non-copiable would allow competitors to emulate a popular interface if the code was not substantially similar to the original's. Judge Keeton concluded that the law could not possibly allow such behavior. \textit{Lotus Dev. Corp. v. Paperback Software Int'l}, 740 F. Supp. 37, 65-68 (D. Mass. 1990).
\textsuperscript{255} Menell, \textit{supra} note 159, at 1070.
\textsuperscript{256} \textit{Id.} at 1064.
unwarranted protection of ideas and functions.\textsuperscript{257} The interpretations given to copyright law in the \textit{Whelan} era granted creators unjustified monopoly power over ideas and processes. Most of the improvements that received protection under copyright would never have satisfied the novelty and non-obviousness requirements of patent law. These decisions essentially allowed developers to circumvent patent law.\textsuperscript{258} When this "forced adaptation"\textsuperscript{259} of copyright law to computers insulates ideas that should be disseminated to the public, other developers cannot offer the user-friendly products that consumers desire. Thus Menell advocates a specific additional requirement that protection not extend to any "user-friendly" feature designed solely for that purpose.\textsuperscript{260}

In failing to lay down firm principles for circuit courts to follow, the Supreme Court has left those courts free to adopt whatever policies they see fit, only limited by other well-established principles. Thus, the debate over the proper scope of copyright protection remains murky, and threatens to become even murkier.

\section*{V. CONCLUSION}

Computers continue to grow in popularity. Thus, demand grows for more sophisticated software to help users perform everyday tasks. Just like other creators and inventors, software programmers and developers want to reap the rewards of their endeavors and prevent others from profiting by essentially stealing the creators' ideas and products. Thus, it is important to define the proper form of protection for software.

The First Circuit, in \textit{Lotus v. Borland}, took the initial step toward striking a balance between protection for the developers and dissemination for the good of the general public. That court

\textsuperscript{257} \textit{Amicus Brief, supra} note 13, at 33 ("The very fact that such functional ideas are so valuable explains why copyright law, with its low standards for obtaining protection and its long duration of exclusive rights, will not protect them"). Although patent has downfalls and could also stifle the advancement of software, the requirements are more stringent for such protection, and the duration is much shorter. However, duration is far less of a concern because software becomes virtually obsolete so quickly.

\textsuperscript{258} Menell, \textit{supra} note 159, at 1096-98.

\textsuperscript{259} Rice, \textit{supra} note 195, at 1136.

\textsuperscript{260} Menell, \textit{supra} note 159, at 1099.
recognized the key issues with respect to copyright protection for software and attempted to set forth a sound analysis to determine when copyright should apply. In doing so, the First Circuit reached the correct result, but not necessarily for the right reasons. The Supreme Court appeared to recognize the necessity for establishing those right reasons when it granted certiorari to hear the case.

Alas, the Supreme Court failed to move beyond mere recognition. Instead of taking a bold step forward into the computer age, the Court left copyright law in the same state of confusion as it found it. The circuit courts therefore continue to have no guidance as to what policies and doctrines should apply. However, there is some good news for the circuits: no Supreme Court precedent will impede them when they adopt whatever policies they choose.

Bradley W. Grout