A COHERENT THEORY FOR THE COPYRIGHT PROTECTION OF COMPUTER SOFTWARE AND RECENT JUDICIAL INTERPRETATIONS

Dennis S. Karjala*

I. INTRODUCTION

This Article will focus on recent developments in the copyright protection of computer software. In order to go beyond mere description and evaluate these developments, however, I must supply an analytical framework against which specific legislative schemes and judicial decisions can be tested. A given analytical framework, in turn, either explicitly or implicitly implements a particular vision of optimal social policy. I therefore offer such a vision of social policy as well.

I summarize this vision of social policy right at the outset: the problem of software protection under copyright law cannot be meaningfully addressed by viewing copyright as a stand-alone statute. Rather, intellectual property law must be viewed as a coherent whole, with copyright, patent, trade secret, unfair competition, and trademark law each playing different but interrelated roles. Of particular importance for software protection are the relative roles of patent and copyright. Resolution of practically every important problem of software protection will vary depending on how one answers this fundamental question: why did we place computer programs—technologically functional works—under copyright instead of relying on the traditional mode of intellectual property protection for technology, namely, patent law?

My answer to this question starts from the observation that protection of the fruits of intellectual creativity is the object of both patent and copyright law; therefore, that many aspects of computer programs are highly creative cannot serve as a basis for placing programs under copyright instead of patent law. I believe, rather, that computer programs were brought under the copyright umbrella because they were thought to be vulnerable to misappropriation through cheap, fast, and accurate electronic copying; because even costly-to-make programs were vulnerable but often would not qualify for patent protection; and because copyright was a convenient at-hand tool for achieving the desired result. The implications of this answer to the "why copyright for

* Professor of Law, Arizona State University College of Law; Irving Younger Visiting Professor of Law, University of Minnesota Law School, 1997-98.
software” question are the subject of this Article. One implication, in particular, is that we should protect under copyright only those functional computer program elements that are vulnerable to misappropriation (primarily code), and leave other program elements—even highly “creative” elements—to their fate under the traditional schemes for protecting technology, namely, patent and trade secret law.

Copyright’s basic mission demands tradeoffs between two conflicting policy goals: optimizing creation incentives through rewards to authors and optimizing public access to and use of desirable works. If we underprotect, we might not get as many works as we desire, and those works that are created might be made available only under strict conditions of access. If we overprotect, on the other hand, we might not supply much increased incentive, but new authors will be inhibited in what they can use from the existing base in creating new works. For technological works like computer programs, which are new to copyright, coherent analysis requires an even clearer explication of the underlying policy copyright protection seeks to implement. The failure of both Congress and the courts to develop a consistent, clearly articulated policy basis for the copyright protection of computer programs is the primary source of the continued confusion in the case results and the need for ongoing public discussion over both policy and methods of implementation.

The debate is not over copying of code for resale, or even over private copying of code for the purpose of using the program to accomplish the function intended by its author. Copying of this type is copyright infringement, absent fair use, and copyright law works at least as well as any legal ban against copying can work to prohibit that kind of piracy. The debate, rather, is over what other aspects of program technology beyond literal code should be protected by copyright. The following three problems, in particular, have been the focus of much of the academic, as well as judicial, attention: the scope of copyright protection in a program; the copyright protection of interfaces, especially user interfaces; and the reverse analysis (or reverse engineering) of programs.

Each of these three issues raises the fundamental question of the extent to which the traditional distinctions between copyright and patent can and should be maintained in the digital age. As a basic policy matter, the question has, of course, no single right or wrong logical resolution. Nevertheless, it is a question that is fundamental to our entire intellectual property protection scheme. It should therefore be addressed directly, with arguments based on intelligible axioms and logic. Unfortunately, this policy issue is easily hidden in the abstract
analysis and terminology of copyright applied, often superficially, to specific factual settings.

We must always bear in mind that failure to protect under copyright does not necessarily mean a complete absence of protection. Many program elements and features can be protected under patent or trade secret law. Moreover, we must remember that these other branches, too, draw finely tuned balances between protection and free use. Therefore, even when other branches of intellectual property law also deny protection, there will often be a valid reason based on a judgment that protection would result in more social harm than benefit. As a result, we must be cautious in allowing copyright to reach outside its traditional realm to protect functional elements of works that are denied protection under the branches of intellectual property law designed and honed over the years for application to such elements.

This Article attempts first to establish a frame of reference by articulating a detailed view of the relationship between patent and copyright and the implications of this view for the copyright protection of computer programs. As outlined above, it finds the policy basis for protecting computer programs under copyright not in the artistic creativity that goes into program production, but rather in the necessity of protecting expensive-to-create-but-easy-to-copy digital works from piracy (or misappropriation). The Article then moves on to judicial developments in the three most important problem areas in the application of copyright to the protection of computer programs, with emphasis on the recent cases: the scope of copyright protection in programs, the copyright protection of interfaces, especially user interfaces, and reverse engineering of programs.\(^1\)

---

II. FUNCTIONALITY AS THE DISTINCTION BETWEEN PATENT AND COPYRIGHT

The fundamental difference between traditional patent and copyright subject matter is simple: patent protects creative but functional invention; copyright protects creative but nonfunctional authorship. "Creativity" does not distinguish between the two regimes. Both protect intellectual creativity. Neither demands or protects, in general, more creativity than the other. Neither regime protects, in general, a different kind of creativity from the other.  

2. This section is based on Reverse Engineering and Professor Miller, supra note 1, at 976-83, which in turn was based on Dennis S. Karjalta, Reverse Engineering of Computer Programs and the Challenge to the Patent/Copyright Boundary, presented at the Stockholm Congress of the International Association for the Advancement of Teaching and Research in Intellectual Property, Stockholm School of Economics, Stockholm (August 17-19, 1993); see also Brief Amicus Curiae of Professors Peter S. Menell and Dennis S. Karjalta in Support of Respondent, Lotus Dev. Corp. v. Borland Int'l, Inc., 49 F.3d 807 (1st Cir. 1995), aff'd by an equally divided Court, 116 S. Ct. 804 (1996) (No. 94-203); [hereinafter Lotus Amicus Brief].

3. Very few commentators on the question of protecting computer programs under copyright attempt to address the implications of patent law. Professor Friedman has looked in this direction and suggested a somewhat different approach to the patent-copyright boundary. He argues that the copyright paradigm is applied when copying is easy and easily recognized and independent invention is unlikely, while the patent paradigm is applied when copying is expensive and hard to recognize and independent invention is likely. See David D. Friedman, Standards as Intellectual Property: An Economic Approach, 19 U. DAYTON L. REV. 1109, 1118 (1994). This approach essentially treats copyright as a whole as an antimisappropriation law and leads to the same result for computer program code as the approach I suggest. If we were free to create new antimisappropriation statutes at will, we might well decide to protect certain types of functional works under a copyright-like statute, but for a shorter term or with some form of compulsory licensing so that the work in question could be improved by others with lower transaction costs. The Semiconductor Chip Protection Act is one case in which we have adopted sui generis legislation for a specific type of intellectual property. I, too, have suggested the possible adoption of a third paradigm based on antimisappropriation, under which statutes would be tailored to the characteristics of the works protected and the distinctions between the different types of subject matter would be multidimensional. See Misappropriation as a Third Intellectual Property Paradigm, supra note 1; see also Rochelle C. Dreyfuss, A Wrong's Approach to Information Products: Misusing Copyright and Patent Into a Unitary Theory of Intellectual Property, 1992 SUP. CT. REV. 195, 221-29. For an even more ambitious approach to sui generis legislation, see J.H. Reichman, Legal Hybrids Between the Patent and Copyright Paradigms, 94 COLUM. L. REV. 2431 (1994); Pamela Samuelson et al., A Manifesto Concerning the Legal Protection of Computer Programs, 94 COLUM. L. REV. 2308 (1994) [hereinafter Software Manifesto].

In fact, however, we rarely write new statutes to deal with new kinds of works. At present, we have only two general statutes for the protection of the fruits of intellectual creativity—patent and copyright. The decision to place a type of work under copyright, for example, brings with it not only protection against literal copying, but also prohibitions on compulsory licensing, a very long period of protection, and an uncertain scope of protection against even nonliteral copying, not to mention several hundred years of copyright jurisprudence. For that reason, the courts have not divided patent and copyright subject matter according to the scheme suggested by Professor Friedman. The accounting system and implementing forms at issue in Baker v. Selden, 101 U.S. 99 (1880), were easy to copy and copying could be easily recognized. Nevertheless, copyright protection was denied both for the system and for its "user interface," the implementing forms. The refusal of the Supreme Court to recognize state-created copyright-like protection for the "photocopying" of boat hulls also shows that there is more to the actual implementation of the patent-copyright distinction by the statutes and the courts than is captured by the features suggested by Professor Friedman. See Bonito Boats, Inc. v. Thunder Craft Boats, Inc., 489 U.S. 315 (1989).
In taking "functionality" as the fundamental distinction between patent and copyright, it is crucial to establish a careful definition of the term. A number of commentators have argued that functionality is not a barrier to copyright protection by equating the term "functional" to "useful." This equation of usefulness with functionality, however, leaves no distinction between patent and copyright subject matter and raises the question of why we should have two very different statutory schemes for protecting the same thing.

The underlying social policy goals of patent and copyright are similar. The purpose of both is to draw a balance between, on the one hand, providing an incentive for the creation of works desired by society and recognizing in some fair and just way the efforts of their creators and, on the other hand, ensuring a broad public domain that permits later inventors and authors to build on the existing foundation to advance technology and culture for the overall benefit of society. Nevertheless, these two great statutes effect their social policy balances in very different ways. Patents must be narrowly claimed and protection is narrowly limited to the claim. Patents must be approved by administrative authorities, must involve a "nonobvious" and not simply a normal engineering advance, and remain valid for twenty years from filing. Copyrights, on the other hand, come into existence automatically with no requirement that rightsholders specify precisely which aspects of their works are protected and which are not. Moreover, the copyright-protected work need only be the intellectual product of its author, with at most minimal artistic creativity, and protection continues for roughly seventy-five to one-hundred years. In sum, patents are narrower than copyrights, harder to get, and persist for a shorter period. Functionality, properly defined, is the key to understanding why these very different statutory styles, both aimed at protecting the fruits of intellectual creativity, have mutually coexisted for such a long time.

In fact, it is clear that traditional copyright law, following the dictates of *Baker v. Selden*, has eschewed the protection of function, provided the meaning of the term "function" is understood in a precise way.

141 (1989). It is true that that basing subject matter distinctions on functionality is not perfect, either. See infra text accompanying notes 33-38. Nevertheless, it is impossible to capture the actual distinction between patent and copyright, as the two statutes are applied today, without regard to functionality.


5. 101 U.S. 99 (1880).
Copyright does, of course, protect many works that are useful to human beings. Maps enable us to go from one place to another; recipes tell us how to bake cakes; accounting books explain how to implement a particular system of accounting. All of these works are copyright protected. They are not, however, “useful articles” within the definition in the Copyright Act, which provides that “[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 . . . .”

This definition was adopted in an effort to deal with the problem of industrial design by excluding utilitarian functions other than to inform or portray an appearance, from copyright protection and by excluding even expressive aspects of useful articles to the extent they are not separable from the utilitarian functions. However, it also represents a statutory description of the kind of usefulness or functionality that has always been excluded from copyright protection, as opposed to the usefulness inherent in maps and recipe books that have long been a part of copyright.

The code book cases do not supply precedent for the copyright protection of function under this definition. Many of these cases determined the existence of a copyright from reasoning or standards that are no longer tenable. In American Code Co. v. Bensinger, for example, it appears from the example given in the opinion, that the plaintiff created a set of five-letter code combinations by simply incrementing letters of the alphabet in the same way that one increments numbers. Although the court found the set to be copyright protected, the case would raise serious questions of originality under today’s standards. The Bensinger court, moreover, based copyright protection solely on Jeweler’s Circular Co. v. Keystone Pub. Co. Jeweler’s Circular and the “sweat of the brow” theory of originality for which it became the basis, were thoroughly repudiated by the Supreme Court in Feist. The question of copyright protectability of codes was not argued in Hartfield v. Herzfeld, which in any event was heavily influenced by the sweat of the brow reasoning of the Second Circuit precedents binding the district

8. See, e.g., Hartfield v. Peterson, 91 F.2d 998 (2d Cir. 1937); American Code Co. v. Bensinger, 282 F.2d 829 (2d Cir. 1922); Hartfield v. Herzfeld, 60 F.2d 599 (S.D.N.Y. 1932); Reiss v. National Quotation Bureau, Inc., 276 F. 717 (S.D.N.Y. 1921).
9. 282 F.2d 829, 833 n.1 (2d Cir. 1932).
10. 281 F. 83 (2d Cir. 1922).
12. 60 F.2d 599 (S.D.N.Y. 1932).
court. The Hartfield v. Peterson\textsuperscript{14} court found copyright protection for the collection of code words as a compilation and came to the astounding conclusion that copying of anything from a protected compilation was infringement.\textsuperscript{15}

In any event, the code book cases are not authority for the copyright protection of functionality, provided the term "function" is understood properly. Like Reiss v. National Quotation Bureau, Inc.,\textsuperscript{16} the most often cited of the code book cases, these decisions hold, at most, that a book of meaningless code words may be copyright protected. That does not make the works functional in the sense that distinguishes patent from copyright. Under today's statutory definition of a useful article, the issue is whether such a list of code words has an intrinsic utilitarian function other than to portray the words included in the list or to convey information. Such meaningless words, by definition, do not in themselves convey information; their only apparent function is simply to portray themselves when meaning or syntax is added to make a secret code. Had the code books used newly created hieroglyphic-type characters, few would doubt the copyright protectability of the characters as pictorial works or assert their status as a useful article, because the only function of the characters would be to portray themselves, whether or not the user adds meanings or syntax. The code words in Reiss and the other code book cases portray themselves as letter groups rather than pictures, but they have no other utilitarian function, intrinsic or otherwise. Consequently, they are not functional in the sense that distinguishes patent and copyright subject matter.\textsuperscript{17}

Thus, a work is functional for the purpose of distinguishing between patent and copyright subject matter if it performs some utilitarian task other than to inform, entertain, or portray an appearance to human beings.\textsuperscript{18} As discussed above, usefulness in the ordinary sense does not

\textsuperscript{13} Id. at 600 ("[I]t must be recognized that the plaintiff's basic work . . . was the result of years of labor . . . .").

\textsuperscript{14} 91 F.2d 998 (2d Cir. 1937).

\textsuperscript{15} See id. at 1000. "When the statute allows a compilation to be copyrighted, it seems clear that no one can copy phrases or sequences which are original with the author or appropriate any other part of the copyrighted work, whether that part is in the public domain or not." Id.

\textsuperscript{16} 276 F. 717 (S.D.N.Y. 1921).

\textsuperscript{17} This was obvious to Judge Hand in Reiss: "Baker v. Selden . . . is too foreign to the case at bar to deserve comment." Id. at 719.

\textsuperscript{18} Functionality, as so defined, does not exhaust the distinction between patent and copyright subject matter. A "system," for example, is excluded from copyright protection under § 102(b), although it may be patent subject matter as a "process." Many systems or processes are conceptual algorithms that inform human beings how to do something and are not self-executing. They are, then, not directly functional within the definition offered in the text. Elaboration of this important point, however, must await another article. It might be noted that Professor Cohen descriptively applied the label "functional" to systems, processes, and methods of operation to the extent they are excluded from copyright protection.
make a work functional under this definition; a map under this
definition is nonfunctional, even though it is often quite useful in
traveling from A to B. A doll and a toy airplane are nonfunctional even
though either may be useful in keeping children productively occupied.
A picture is nonfunctional even though it may be useful for decorating
a house or office. A building, on the other hand, is functional,
notwithstanding that part of its design may be to portray an aesthetic
appearance to human beings, because it also protects from the wind and
rain. A computer program in object-code form is functional because it
causes a computing machine to operate so as to achieve a certain
result. All program-to-program interfaces and protocols are functional
under this definition, in view of their inherent utilitarian function of
allowing interoperability between computers and the programs that
govern computer operations. Many computer program user interfaces,
such as the Lotus 1-2-3 user interface at issue in Lotus Development Corp.
v. Borland International, Inc., are also functional in the sense used here,
because they have the intrinsic functional goal of permitting users to
engage in such operations as inputting and manipulating data in a fast,
efficient, and easy-to-master manner.

The enormous differences between the patent and copyright
protection schemes are largely attributable to patent’s primary role as
the protector of functional works, as here defined, and copyright’s
primary role as the protector of nonfunctional works. The policy basis

under § 102(b). Julie E. Cohen, Reverse Engineering and the Rise of Electronic Vigilantism: Intellectual Property Implications of “Lock-Out” Programs, 68 S. CAL. L. REV. 1091, 1145-46, 1200 (1995). Professor Cohen’s conclusions do not fundamentally differ, however, from those reached under the analysis of this Article. In particular, she argues strongly that the elements of computer programs protected by patent must be mutually exclusive from those protected by copyright. See id. at 1199.

19. That computer program code is functional, as defined in the text, does not imply that it is
unprotected by copyright. Despite whatever else we may argue about with respect to computer software,
everyone agrees that, at a minimum, Congress has brought literal code under copyright protection.
Nevertheless, the functionality of code, together with the traditional functional-nonfunctional division of
labor between patent and copyright, has important implications for the degree of copyright protection we
recognize for nonliteral elements of programs and for interfaces. See infra notes 53-66 and accompanying
text.

20. 49 F.3d 807 (1st Cir. 1995), aff’d by an equally divided Court, 116 S. Ct. 804 (1996).
21. The Federal Circuit’s important decision in In re Lowry, 32 F.3d 1579 (Fed. Cir. 1994),
recognizes as patent subject matter “an efficient, flexible method of organizing stored data in a computer
memory.” Id. at 1580. Quoting from In re Bernhart, the Federal Circuit disallowed a “printed matter”
rejection where the invention required that information be processed by a machine rather than by humans.
Id. at 1583-84 (quoting In re Bernhart, 417 F.2d 1395, 1399 (C.C.P.A. 1969)). This precisely the
distinction proffered herein for distinguishing patent from copyright subject matter. The Lowry opinion also
shows the difficulty of relying on terms like “expressiveness” or “creativity” prior to making the patent-
copyright breakdown. Id. at 1580. The court noted that the prior art disclosed data models that were
either “functionally expressive” or “structurally expressive” and that Lowry’s invention sought “to optimize
both structural and functional expressiveness.” Id. Thus, it is insufficient simply to conclude that something
for these differences between the two protection schemes is the social desirability, indeed necessity, of allowing later technological creators—creators of functional works—to build on and improve, often in small ways, the earlier works of others. Technological improvements are often substantially similar to the products they improve and would infringe if the copyright paradigm were applied. Yet, unless they adopt all of the elements of a patent claim, or their substantial equivalents, they do not infringe. Current authors of copyright-protected works too, build on public domain works and unprotected elements of protected works, but in taking from the latter, the broad scope of copyright protection forces them to rely only on the more general or abstract features.

Later inventors thus can apply a “ground up” approach to reliance on earlier protected works. Once they are outside all patent claims, they are safe. Later authors, however, must adopt a “top down” approach and take a serious risk of being held liable for infringement as their reliance becomes more detailed. The functionality-nonfunctionality distinction between patent and copyright explains these different approaches to infringement. The social utility of allowing subsequent authors to make minor variations on a copyright-protected novel, for example, is minimal. For works of fiction, art, and music, variety is the spice of both legal and real life. We prefer to have one hundred different war novels than one hundred versions of War and Peace that differ only in their final chapter. Consequently, the broad scope of copyright protection for novels and the long period of its duration fulfills the goal of recognizing the author’s creativity without unduly hindering later authors or depriving society of desirable works.

Technology, however, improves incrementally, as later inventors add a bell or a whistle to an earlier invention to make it more desirable or useful to consumers. Many improvements on existing products are rather straightforward, or “obvious” in the sense of patent law, and they are given no intellectual property protection once they are released to the public. These products often show at least as much intellectual creativity as many copyright-protected works, but their creators have a monopoly only for the period that is required for competitors to

is expressive to make it copyright subject matter. Much patent subject matter can also be labeled expressive or creative. Consequently, although functionality alone does not wholly solve the problem of distinguishing between patent and copyright subject matter, see supra note 18 and infra notes 33-38 and accompanying text, any coherent analysis must at least begin with functionality.

22. See New Protectionism, supra note 1, at 39.

23. See Kepner-Tregoe, Inc. v. Carabio, 203 U.S.P.Q. 124, 131 (E.D. Mich. 1979). “[T]here is no societal interest in many variants on a single theme or plot, nor is there the likelihood that by extending broad protection, entry to the market for literary works will be foreclosed.” Id.
recognize the value or popularity of the improved product, figure out its "secret," if any, and gear up for production and marketing. In the case of technological products, we have drawn the social policy balance at a different point than for traditional works of authorship because we believe that to grant intellectual property rights in ordinary engineering advances would hinder the development of more products than it would encourage. Hence the "nonobviousness" requirement of patent law, its shorter term, and its requirement for an explicit statement of the claimed invention.

That patent protects functional works is expressly reflected in the "usefulness" requirement that has always been a part of patent law. Copyright, however, tends to define its subject matter through lists of the types of works protected, such as "literary works," "musical works," and "pictorial, graphic, and sculptural works." If something fit within one of these categories, or was similar to works within these categories, copyright protection was often recognized without any inquiry into whether the work was functional. Nevertheless, at least before computer programs appeared on the scene, the vast majority of works protected by copyright were useful only in the sense that they entertained (by presenting an appearance or sound) or informed human beings. They could be put to use only through human intervention and interpretation. Thus, the functional-nonfunctional distinction between traditional patent and copyright has long existed, but from the copyright side, it has been only implicit in the statute and the cases.

United States law is nevertheless among the clearest in seeking to maintain the distinction and to limit copyright protection in functional works. First, under the idea-expression distinction, courts narrow the "expression" found in functional or utilitarian works, and afford these works only "thin" protection.

24. 17 U.S.C. § 102(a) (1994) (stating that copyright subsists in "original works of authorship" and going on to include enumerated categories within the term "works of authorship").

25. Thus, a recipe for baking a cake might be very useful if one wishes to have dessert after dinner, but the recipe itself is not functional because it does not bake the cake. Only the human being reading the recipe does that, and he or she is free to make variations as the mood strikes, over which the recipe has no control.


27. E.g., Continental Cas. Co. v. Beardsley, 253 F.2d 702 (2d Cir. 1958). The legal forms involved in this case are functional within my definition because, in addition to providing information, they establish legal relationships among the parties who execute them.
Baker v. Selden\(^2\) and its progeny have taught that copyright's reproduction right in a particular work is subordinated to free competition when its assertion would otherwise inhibit competition in the market for non-copyright-protected products. The copyright in a book, for example, does not extend to systems and procedures that are described in the book.\(^2\) Moreover, third parties have always been allowed to copy, say, an unprotected three-dimensional boiler or dress,\(^3\) even though the reverse engineering of any complex product normally involves making intermediate two-dimensional designs based on the product being copied.\(^4\) These cases stress the importance of insuring that copyright, with its low threshold of eligibility for protection, does not displace the more stringent requirements for protection of functional works under patent law.\(^5\)

To be sure, to the extent that industrial designs and architectural works have been protected under copyright, part of this patent-copyright dichotomy based on functionality may be said, at times and in varying ways in different countries, to have broken down, even before

\(^2\) 101 U.S. 99 (1880).

\(^3\) See, e.g., Affiliated Enters., Inc. v. Gruber, 86 F.2d 958, 961 (1st Cir. 1936) (holding that the promotional scheme is protectable only by patent, regardless of how good it is or how costly it was to develop); Affiliated Enters., Inc. v. Gantz, 86 F.2d 597 (10th Cir. 1936) (similar promotional scheme); Brief English Sys., Inc. v. Owen, 48 F.2d 555 (2d Cir. 1931) (noting that only patent is available to protect a new system of shorthand); Arica Inst., Inc. v. Palmer, 761 F. Supp. 1056 (S.D.N.Y. 1991) (holding method of describing psychological traits not protected); Kepner-Tregoe, Inc. v. Carabio, 203 U.S.P.Q. 124, 130 (E.D. Mich. 1979) (finding that Kepner-Tregoe had no monopoly on any pedagogical technique involved in a management training program); see also Pamela Samuelson, Computer Programs, User Interfaces, and Section 102(b) of the Copyright Act of 1976: A Critique of Lotus v. Paperback, 6 HIGH TECH. L.J. 209, 226-27 & n.73 (1992).


\(^5\) See Sega Amicus Brief, supra note 26, at 19-20, 33 JURIMETRICS J. at 157 (the primary author of this portion of the brief was Professor Reichman). Therefore, when new designs are drawn up based on the three-dimensional product, they are likely to be substantially similar to, and indirectly taken from, the plaintiff's copyright-protected design documents.

\(^6\) See Applied Scientific Know-How, supra note 26, at 692. Baker itself emphasizes this point: “To give to the author of the book an exclusive property in the art described therein, when no examination of its novelty has ever been officially made, would be a surprise and a fraud upon the public. That is the province of letters-patent, not of copyright.” 101 U.S. at 102.
the advent of computer programs.\textsuperscript{33} In the Copyright Act of 1976, the United States sought to continue the subordination of copyright to function for industrial designs through its separability test for pictorial, graphic, and sculptural (PGS) works,\textsuperscript{34} but the recent addition of the separate class of "architectural works" to the statutory list of copyright-protected works means that building designs are no longer subject to the separability test.\textsuperscript{35} Presumably, the long-standing doctrine of \textit{Baker v. Selden} will continue to limit or deny copyright protection to functional aspects of buildings, but only time and the courts will tell us. In addition, over the years courts have afford copyright protection to occasional isolated examples of functional works, in the sense used herein, although these courts rarely evince an understanding that they have actually brought functionality under copyright.\textsuperscript{36}


\textsuperscript{34} 17 U.S.C. § 101 (1994) (definition of pictorial, graphic, and sculptural works). This definition states that the design of a useful article is a pictorial, graphic, and sculptural (PGS) work only to the extent that it incorporates PGS features that are separately identifiable from and capable of existing independently of the utilitarian aspects of the article. See id. § 101. The separability test has caused considerable problems in the courts, but for the purposes of this Article it is enough to note that the effort was made to subordinate copyright to function for this class of works.

\textsuperscript{35} Professor Ginsburg, too, has argued that "functionality" is no bar to copyright protection. Jane C. Ginsburg, \textit{Four Reasons and a Paradox: The Manifest Superiority of Copyright over Sui Generis Protection of Computer Software}, 94 COLUM. L. REV. 2559, 2566 (1994). Most of her examples, such as architectural plans, choreography, musical scores, musical editions, and stage direction, are not functional in the sense used herein to distinguish patent from copyright, in that they simply instruct human beings how create or make palpable the work in question. See id. However, she also offers the 1990 amendments to the Copyright Act protecting buildings in support of her assertion. See id. at 2567. She does not offer a general rationale for the copyright protection of function, particularly as herein defined, except to say that it would be "both inaccurate and unfortunate" to extrapolate the useful article definition in connection with PGS works to more general types of works. \textit{Id.} at 2568. She also notes, quite correctly, that the 1980 amendments bringing computer programs under the copyright umbrella necessarily made some inroads into § 102(b). See id. at 2569-70; see also infra notes 44-47 and accompanying text. She does not, however, offer any rationale for distinguishing patent and copyright subject matter, nor does she appear to consider that the scope of potential patent protection might have some bearing on the recognized scope of protection under copyright.

\textsuperscript{36} A clear example is that of standardized test questions that seek to measure intellectual or psychological traits from human responses to the questions. \textit{See Educational Testing Serv. v. Katzman}, 793 F.2d 533 (3d Cir. 1986); \textit{Applied Innovations, Inc. v. Regents of the Univ. of Minn.}, 876 F.2d 626 (8th Cir. 1989). Even those skeptical of the value of some of these tests will agree that their purpose is to measure real-world phenomena, and therefore they have a utilitarian purpose (measurement) other than to entertain or inform. Therefore, they are functional in the sense used in this Article. Some blank forms and classification schemes also receive protection notwithstanding their functionality as systems for presenting information in a convenient or otherwise more useful form. \textit{See K ey Publications, Inc. v. Chinatown Today Publ'g. Enters., Inc.}, 945 F.2d 509 (2d Cir. 1991) (yellow page classification scheme); Kregos v. Associated Press and Sports Features Syndicate, Inc., 937 F.2d 700 (2d Cir. 1991) (permitting copyright protection of blank forms used for baseball statistics when there was a minimal level of creativity in the compilation of the blank forms). See generally Dennis S. Karjala, \textit{Copyright and Misappropriation}, 17 U.
The functionality-nonfunctionality distinction between patent and copyright is therefore not historically perfect. It is surely correct to say, however, that copyright has in the main eschewed protection of function, and judicial and legislative efforts to allow copyright to control markets for functional products have almost always been met with fierce debate. Patent and copyright have evolved alongside one another over a period of several hundred years, and yet the protection schemes are very different. There must be some reason other than inertia that the two have coexisted so long. Only functionality explains this dichotomy in any fundamental way. To the extent the distinction between the two primary intellectual property regimes based on functionality is not quite descriptively correct, I simply offer as a normative proposition that, with its low threshold of eligibility and long protection period, copyright should not be allowed to encroach upon the traditional domain of patent to protect functional aspects of works absent a clearly articulated social policy basis. This, then, is the fundamental policy question that should be expressly addressed by those seeking broader copyright protection for software: why should we protect creative aspects of software technology under copyright while leaving all other creative technological advances to their fate under patent or trade secret law?

In fact, there exists a social policy basis for protecting computer programs, notwithstanding their functional nature, under copyright. That policy, however, also clearly indicates how copyright protection in programs and their interfaces should be limited to avoid upsetting the careful balance between copyright and patent that has evolved over

DAYTON L. REV. 885, 922-26 (1992). But protecting functionality is not the only mistake courts sometimes make in applying copyright to specific fact patterns. A model for managerial decisionmaking for use in training managers is not directly functional, see supra note 18, but it is the kind of thing that should be excluded from copyright protection under Baker v. Selden and § 102(b) as a system or process. Nevertheless, the court held that this type of system was copyright protectable in Kepner-Trego, Inc. v. Leadership Software, Inc. 12 F.3d 527 (5th Cir. 1994).

37. Professor Reichman has described at length the debates over the inclusion of industrial design within copyright. See Reichman, supra note 33. As another example, after the House of Lords decision inBritish Leyland Motor Corp. v. Armstrong Patents Co., 1986 App. Cas. 577 (appeal taken from C.A.), the British copyright statute was amended to deny copyright infringement in making an article to the design of a design document or model and to include "must-fit" and "must-match" exceptions to the design right in an article. Copyright, Designs and Patents Act, 1988, §§ 51 & 213(3)(b).

38. Professor Lunney apparently does not believe that functionality explains the differences between patent and copyright subject matter, because he has criticized the reasoning of commentators (including this author) who "have articulated a number of differences between patent and copyright law that they believe make patent better suited to the protection of useful articles." Glynn S. Lunney, Lotus v. Borland: Copyright and Computer Programs, 70 TUL. L. REV. 2397, 2420 n.70 (1996). His analysis does not attempt to explain just why we have two so very different statutes for the protection of the fruits of intellectual creativity. If patent law is not "better suited to the protection of useful articles," what is it better suited for?
time. The short answer is that the program copyright should protect only the literal code and mechanical or electronic translations of code. Higher level aspects of program structure and design should be considered patent subject matter and protectable under patent law if they meet the stricter patent standards. Fanciful aspects of user interfaces, such as video game characters, should be copyright protected as traditional pictorial or graphic works, independent of the copyright in the program that generates them on a screen or other input-output device. Other aspects of user interfaces must also be protected, if at all, either by an independent copyright as a pictorial, graphic, audiovisual, musical, literary, or similar traditional work, or as a nonobvious technological advance under the patent standards. The next section of this Article develops these conclusions in more detail.

III. THE TECHNICAL AND POLICY BASES FOR SOFTWARE COPYRIGHTS

A. The Underlying Policies

Computer software is technology—the means by which computing machines are caused to perform their jobs. Computer programs sequentially set the switches inside the machine in such a way that the results can be interpreted by human beings as “information processing.” As outlined above, the traditional intellectual property protective mode for technology is patent. Computer programs, communication protocols, hardware-to-software and software-to-software interfaces (nonuser interfaces), as well as many user interfaces are intrinsically functional in the sense herein defined. They cause computing machines to achieve their results and either serve as, or open and shut, the doors and windows by which programs are used or interact with each other or with hardware. Thus, they have utilitarian purposes other than to inform or entertain human beings. In fact, except possibly for the purpose of setting “traps” to detect copying, everything in a program and its nonuser interfaces (as well of much of its user interface) is there to achieve a functional purpose, such as accuracy, speed of processing,

39. The following section is based on Recent International Developments, Part I, supra note 1, at 13-20, and Reverse Engineering and Professor Miller, supra note 1, at 983-95.

40. Because user interfaces, such as those for video games, may include fanciful, nonfunctional aspects, it is often important to distinguish them from other interfaces and communication protocols. However, the term “communication protocols, hardware-to-software and software-to-software interfaces” is long and clumsy. Henceforth I refer to them with the equally clumsy but shorter term “nonuser interfaces.”
optimal use of available resources, facilitating or hindering access, compatibility with other hardware or software, or ease of learning or use. Why did we suddenly turn to copyright law for the protection of such intrinsically functional works?

Calling programs "literary works" is simply another way of phrasing the question. We could equally, perhaps even more aptly, call them "methods of machine design," in that they take a universal machine and transform it into one that achieves a particular result. The real reason we resorted to a copyright scheme to protect computer programs is that many programs—including programs that are costly and time consuming to develop—are simply the result of technologically straightforward application of well-known programming principles to a well-defined problem. These programs do not meet the requirement of traditional patent law for a nonobvious advance in the art. 41 Yet, once these programs are distributed in object-code form, they can be copied almost without cost in large numbers. Without some form of protection, we should expect that they would be underproduced. Because the evil to be avoided was thus slavish copying, especially slavish electronic copying, because copyright protects at least against that, and because computer programs formally fit the broad definition of a literary work under copyright law, 42 it became a natural candidate for the protection of programs, notwithstanding their inherent functionality. 43

To the extent that program code is now copyright protected, Congress has formally and expressly indicated that a functional work

41. At least one commentator has argued that implementing a well defined process in a programming language is always obvious and therefore nonpatentable. See Gary Dukarich, Patentability of Dedicated Information Processors and Infringement Protection of Inventions that Use Them, 29 JURIMETRICS J. 135, 160 (1989). Moreover, computer programs "as such" are barred from patentability under Article 32 (Appendix I) of the European Patent Convention, although it is not entirely clear just what "computer programs as such" means. J. Betten, Patent Protection for Software, in ECIS SYMPOSIUM, AN EMERGING WORLD-WIDE CONSENSUS ON SOFTWARE PROTECTION, Brussels, April 27, 1993. The availability of patent protection for noncode aspects of programs, however, has significantly increased in recent years. The Federal Circuit held in Lowry that the printed matter doctrine was improperly applied to reject data structures stored in memory. In re Lowry, 32 F.2d 1579 (Fed. Cir. 1994). This prompted the Patent and Trademark Office to issue new examination guidelines that treat "functional descriptive material" (e.g., data structures and computer programs) as statutory subject matter when encoded on a computer-readable medium. The Bureau of National Affairs, Examination Guidelines for Computer-Related Inventions, 51 PATENT, TRADEMARK & COPYRIGHT J. 422 (Jan. 25, 1996). The effect is largely to change the analytical focus under patent law from the almost metaphysical § 101 subject matter question to the more relevant issue of whether the claimed invention represents a nonobvious advance in the art of software technology.

42. "Literary work[s] are works . . . expressed in words, numbers, or other verbal or numerical symbols or indicia . . . " 17 U.S.C. § 101 (1994).

43. See New Protectionism, supra note 1; see also Dennis S. Krajicek, Copyright Protection of Computer Software in the United States and Japan, 13 EUR. INTELL. PROP. REV. 195 (Part I) & 231 (Part II) (1991). Another advantage of copyright is the immediate international nature of the protection under the copyright treaties. See id. at 196.
should come under the copyright umbrella. To determine, however, what else besides code is protected by the program copyright, we must inquire more deeply into the congressional purpose. Because Congress itself has said nothing in either the statute or any of the statute's legislative history that clarifies exactly what it intended to cover in expressly recognizing the copyright protection of computer programs, we must glean congressional intent from other sources, from the statutory language, and from a sensible interpretation of statutory policy. It is clear that the National Commission on New Technological Uses of Copyright Works (CONTU)—established by Congress "to assist the President and Congress in developing a national policy for protecting both the rights of copyright owners and insuring public access to copyrighted works when they are used in computer . . . systems, bearing in mind the public and consumer interest"—was primarily worried about the vulnerability of program code to piracy. The CONTU Report concluded that literal copying of computer programs would dramatically reduce the development costs of second comers and thereby create a disincentive to invest in computer program development. Even if we assume, therefore, that the application of Baker v. Selden and § 102(b) to program code is now more limited than to other technological works—in that the literal code of every program can be said to implement a process or method of operation and yet remains copyright protected—there is no basis for concluding that Congress intended to remove any barriers to the copyright protection of function in other classes of works.

44. Congress amended the Act in 1980 by adding the definition of a computer program in § 101 and replacing former § 117 with a new one, adopting essentially without comment the recommendations of CONTU, the commission established in 1974 to make recommendations concerning computers and copyright. See Robert A. Gorman & Jane C. Ginsburg, COPYRIGHT FOR THE NINETIES 695 (4th ed. 1993). 45. FINAL REPORT OF THE NATIONAL COMMISSION ON NEW TECHNOLOGICAL USES OF COPYRIGHT WORKS 3 (1978) [hereinafter CONTU REPORT]. 46. See id. at 11. Every concrete example of infringement that CONTU offers involves direct and literal copying, such as photocopying of printed source code and one-to-one transcription from magnetic tape or disk to paper. See id. at 22. The Report goes on to state that "[m]ost infringements, at least in the immediate future, are likely to involve simply copying[]" but the problem it predicts for the then-future was a technology that permits programs to be stated orally or permits use of a program without copying. Id. The CONTU Report nowhere refers to copying of "nonsliteral elements," much less copying of user interfaces or other results of programs, as potential infringement. According to Arthur J. Levine, executive director of CONTU, the Commission scoured the country inquiring into the kind of protection needed for computer software and not once did they hear the words "nonsliteral elements," "structure, sequence, and organization," or even "user interface." Remarks of Arthur J. Levine, Technology Workshop on Computer Software Protection, sponsored by the George Washington University School of Engineering and Applied Science and the Kastenmeier Foundation of the University Wisconsin, Washington, D.C., October 1, 1993. 47. The CONTU Report is clear that "[o]ne is always free to make the [computer] do the same thing as it would if it had the copyrighted work placed in it," CONTU REPORT, supra note 45, at 21, and that the fundamental limitations of copyright reflected in Baker v. Selden and § 102(b) would remain in force.
Thus, the policy basis for bringing functional computer programs under the copyright umbrella is the prevention of slavish copying of code that is ineligible for patent protection. The goal is not to protect all creativity found in programs. Other nonpatented works of technology may be freely copied, modified, and improved, no matter how creative the technology is. The creators of such technological developments enjoy only a limited monopoly resulting from the lead time their products have in the market before competitors can produce and market a similar product. Such copying is permitted, even applauded,⁴⁸ not because society devalues technological creativity. Rather, it is permitted because technology advances incrementally, and to forbid such copying would inhibit more creativity than it would engender. Program code arguably requires a different kind of protection, because to allow verbatim electronic copying of programs would reduce even their lead time monopoly almost to zero, and that level of protection seems too little.

The policy goal of software protection under copyright, therefore, should be protection against piracy—methods of copying that too greatly upset the traditional balances of legal and nonlegal protection available for works of technology—and not a wholesale revamping of the intellectual property protection scheme for functional works or for technological creativity.⁴⁹ This, at least, is the conservative

See id. at 18-20. There is no indication in the textual changes to the 1976 Act or the legislative history that Congress intended to extend copyright protection to any functional subject matter other than computer program code. See supra note 46 and accompanying text. To the contrary, the careful definition of “computer program” in § 101, distinguishing between “a set of statements or instructions” (i.e., code) to be used in a computer and the “certain result[s]” (e.g., user interface) such instructions bring about, reinforces this interpretation of the Copyright Act. See infra notes 53-66 and accompanying text.


⁴⁹ This policy goal, which I first developed in New Protectionism, supra note 1, at 36, 87-88, 94-96, is virtually identical to the “market failure” analysis of the authors of the Software Manifesto, supra note 3, at 2365-420. Those authors advocated achieving the policy goal of avoiding market failure (or “piracy” or “misappropriation,” as I phrase it) by a sui generis statute, primarily because they believe that elements of programs other than literal code are also vulnerable to incentive-eroding copying. I remain unconvinced that anyone has made the case that noncode aspects of programs are substantially more vulnerable to piracy than any other types of technological products—which remain protected only by patent and trade secret law. Unless and until that case is made, a proper interpretation of copyright and the other branches of traditional intellectual property law can achieve an essentially socially optimal result. That, in any event, is what this Article attempts to demonstrate.

Professor Lunney, too, has adopted a similar standard (if on very different reasoning):

In determining whether actual copying has gone too far and thereby becomes an infringement, the central issue . . . is whether the copying would enable a competitor to obtain a savings in terms of the time and money required to create and market a competing work significantly greater than the savings we tolerate for creative nonwork products generally.
approach—the approach that least disrupts the traditional intellectual property protection balances, especially the delicate balance between copyright and patent. Possibly there are grounds for affording even broader protection to program technology under copyright. Such grounds, however, do not arise out of traditional copyright law, notwithstanding the formal classification of computer programs as literary works, because traditional copyright did not protect function. Beyond the vulnerability of code to piracy, I have yet to hear any argument that convincingly distinguishes computer programs from other technological products and leads to a broad scope of copyright protection for programs. Because broad copyright protection for technology is the radical, rather than the conservative, deviation from traditional norms, the burden of proving a convincing policy basis for broad copyright protection of functionality should be on those seeking it, at least until Congress has spoken with greater clarity than it has to the present.

Copyright protection of source and object code—the sets of statements or instructions that constitute computer programs—is in itself important and significant protection. Protection of code makes direct copying either for sale or for simultaneous use by others (within, for example, a given business) illegal. Of course, not all instances of program copying can be detected, any more than all copying of other types of copyright-protected works can be detected. No matter what the law says, some people will make copies of useful programs borrowed from their friends without paying. Sales of unlawful copies in any significant amount, however, are readily detectable. Moreover, any employer who distributes illegally made copies of a program among employees in the business runs a serious risk that, at some point, a disgruntled employee will turn him in. Therefore, protection of program code alone is of great importance. It is simply incorrect to say, as did a committee report for the European Parliament during the debates on the European Software Directive, that if reverse analysis were permitted, "legal protection for computer programs would virtually cease to exist."

---

Lunney, supra note 38, at 2435-36. This, of course, is nearly identical to the antipiracy standard that I have been advocating for over ten years: "[D]id the defendant's activities create market advantages not available to persons who reverse engineer in other technological fields?" New Protectionism, supra note 1, at 88; see also id. at 41, 58, 95-96. Professor Lunney, however, seems to think that his standard is general and applies to traditional copyright-protected works as well as computer software. See Lunney, supra note 38, at 2435. I, too, have argued that antismisappropriation reasoning can be usefully applied in other areas of copyright, but it is insufficient to occupy the field. See Karjala, supra note 36, at 917-27.

1997]  

CORPORATE LAW SYMPOSIUM  71

One more point needs recognition: the Copyright Act makes explicit use of its "useful article" definition only to limit the scope of protection in PGS works. The definition of PGS works in § 101 establishes a separability test that the design of a useful article must meet if that design is to qualify for copyright protection; such design is included within the definition of a PGS work only to the extent it incorporates PGS features that can be identified separately from and are capable of existing independently of the utilitarian aspects of the article.\(^ {51}\) Because computer programs are literary rather than PGS works, the separability test applicable to PGS works does not apply to computer programs.\(^ {52}\)

This limitation of the separability test to PGS works, however, does not affect the more general functionality limitations on the scope of copyright protection in computer programs. To apply the separability test to computer programs would defeat the whole purpose of including them within copyright. Because program code is wholly functional, to separate all functional aspects of code from a program would leave nothing at all to protect. Yet it is clear that Congress intended program code, at least, to be protected. That the separability test for PGS works does not apply to programs says nothing, however, about how the larger functionality doctrine that finds its genesis in *Baker v. Selden* applies to noncode aspects of programs. This doctrine, which is partially codified in §§ 102(b) and 113(b) of the Copyright Act, applies to all classes of functional works.

The conservative approach to software copyrights, then, is to preserve as much as possible of the traditional division of labor between patent and copyright law while protecting computer programs against, at a minimum, the kind of misappropriation that CONTU identified as the basis for bringing them under the copyright umbrella in the first place, namely, the direct or indirect copying of literal code. As a policy matter, further protection might be justified on the basis of identified "market failures," that is, where there is a clear danger that expensive-to-create but easy-to-copy nonpatentable features will be underproduced without anticopying protection. Outside of code, however, the parties seeking protection should have the burden of proving that market failure in this sense has occurred or at least is likely to occur. Whether and the extent to which copyright law, as the statute

\(^{51}\) *See* 17 U.S.C. § 101 (1994) (definition of "Pictorial, graphical, and sculptural works").

\(^{52}\) *Cf.* Harper House, Inc. v. Thomas Nelson, Inc., 889 F.2d 197 (9th Cir. 1989) (PGS useful article rule applicable only to PGS works).
is currently drafted, can plausibly be interpreted to cover any such noncode aspects of digital technology is a separate question, but it need not be addressed until functional noncode program elements that are vulnerable to piracy are identified.

B. The Statutory Program Copyright

Under the United States Copyright Act, 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." This definition quite explicitly distinguishes between the "set of statements or instructions" constituting the "computer program" and the "certain result" that the program brings about via execution by the computer. The natural reading of this definition leads to the conclusion that the computer program is, quite simply, the code. This definition has rather important consequences for the degree of protection afforded by the program copyright to so-called nonliteral elements of the program and to the protection of program interfaces, including user interfaces.

It is, of course, well established as a general principle of copyright that the protection of literary works is not limited to the literal language. Equally clear, however, is that the scope of protection varies dramatically according to the type of work, particularly for literary works. Now, given that functionality distinguishes computer programs in a fundamental way from traditional copyright subject matter, including traditional literary works, we would be justified in relying solely on the statutory definition and policy reasoning to determine the appropriate scope of protection, without reference to traditional precedents. Nevertheless, to the extent we seek guidance in case

53. 17 U.S.C. § 101 (definition of "computer program").
54. While Professor Patry has reached diametrically opposite conclusions from those developed herein on nearly every important outstanding issue of software protection under copyright, we are in agreement that the analysis should begin with the language of the statute and the statutory definition of a computer program under § 101. See William F. Patry, Copyright and Computer Programs: It's All in the Definition, 14 CARDOZO ARTS & ENT. L.J. 1, 8 (1996). I will attempt to point out where and how our analyses diverge as we move along.
55. See Sheldon v. Metro-Goldwyn Pictures Corp., 81 F.2d 49 (2d Cir. 1936).
56. Professor Patry and I, for once, seem to be in remarkable agreement both that the variety of traditional literary works leaves no principles of copyright generally applicable to all of them and that, because the end user of a computer program is a machine, programs are fundamentally distinguishable from works intended to be read by humans. See Patry, supra note 54, at 22-23. Fundamental copyright concepts like "idea-expression" and "fair use" are notoriously vague, even in application to traditional copyright subject matter. For traditional artistic, literary, and musical works, however, we can reason by analogy from case law developed over more than two centuries. Once copyright lawyers have read many hundreds of cases, they feel at least a modicum of comfort in advising clients in both planning and in litigation. Analogical copyright reasoning must be applied with great caution, however, to functional works
analogy, we must recognize that under traditional copyright law certain types of literary works receive "weaker" protection than others. In some cases the copyright can be so thin as to be almost nonexistent.\textsuperscript{57} The technical statutory interpretation question, then, is whether computer programs are more analogous under copyright to novels or plays, which receive a relatively broad scope of protection covering nonliteral elements of plot sequence or structure, or to histories, biographies, rule books, fact works, legal forms, and scientific works, in which the scope of protection is much narrower and, indeed, is limited to literal language and quite close paraphrases.\textsuperscript{58} Given the technological nature of computer programs, the question almost answers itself: straightforward application of the traditional copyright precedents for literary works results in a scope of program copyright that is limited to literal code and close paraphrases. Those who assert that the play and novel cases are the relevant authorities\textsuperscript{59} must give policy arguments to support their deviation from the natural choice of literary work authorities—technical and scientific works.

The statutory definition of "computer program" also has important implications for the copyright protection of interfaces generated by computer programs. That definition distinguishes between the program and the certain result that the program brings about. It follows that the certain result effected by a program is not part of the program and therefore is not protected by the program copyright. The code bringing the result about is, of course, protected, but anyone has the right to produce independent code bringing about even an identical result, at least as far as the copyright in the program is concerned. As emphasized by CONTU, "One is always free to make the machine do the same

\textsuperscript{57} See, e.g., Continental Cas. Co. v. Beardsley, 253 F.2d 702, 705 (2d Cir. 1958) (quoting Crume v. Pacific Mut. Life Ins. Co., 140 F.2d 182, 184-85 (7th Cir. 1944)) ("We realize that such a view [allowing similarity of language necessary to the functional use intended as a legal form] leaves little, if any, protection to the copyright owners [sic]; in fact, it comes near to invalidating the copyright.").


\textsuperscript{59} See, e.g., Miller, supra note 4, at 1008, 1020-21 (arguing for treating programs like "other literary works" and comparing system software to works such as Steinbeck's Grapes of Wrath).
thing as it would if it had the copyrighted work placed in it, but only by one's own creative effort rather than by piracy. Even one of the strongest proponents of broad copyright protection for programs agrees that third parties are "free to emulate all external aspects of the program" without infringing the copyright. It is widely recognized that different computer programs can generate identical outputs. Therefore, even identity of unprotected output in no way establishes infringement of the underlying program. If the certain results achieved by program execution are to be copyright protected, they must independently have status as copyright subject matter, that is, as original works of authorship.

    Software interfaces, including user interfaces, are an important part of that certain result which the set of instructions accomplishes. They are the doors and windows through which users, as well as other hardware and software, exchange the signals necessary to interoperate with the program. The program designer must decide what kinds of interfaces are desired and then write code—the set of statements or instructions—to implement those decisions. Of course, the program also achieves other certain results in response to the user's inputs of commands and data, but the precise commands required as well as the types and formats for data that the program can "understand" are all features that are determined by, rather than an inherent part of, the set of statements or instructions that constitute the copyright-protected computer program. Anyone wishing to change an interface, for

60. See CONTU REPORT, supra note 45, at 21.
62. In fact, to the extent that a specific piece of code is the only way of bringing about some aspect of this certain result, one of only a few practical ways of doing so, or an obvious or standard way of doing so from the point of view of an experienced programmer, even that specific piece of code is not protected, and a finding of infringement cannot be based on similarities in these aspects of the code. The CONTU Report states:

    Copyright protection for programs does not threaten to block the use of ideas or program language previously developed by others when use is necessary to achieve a certain result. When other language is available, programmers are free to read copyrighted programs and use the ideas embodied in them in preparing their own works . . . .

    CONTU REPORT, supra note 45, at 20 (first emphasis added). This also follows, of course, from the filtering analysis called for by Computer Assocs. Int'l, Inc. v. Altai, 982 F.2d 693 (2d Cir. 1992).
63. When a user supplies data to a computer program and executes certain commands properly, the user achieves a certain result in the form of a letter on a word processor or a spreadsheet calculation. The sequence of commands and data inputs employed by the user can thus itself be considered a computer program that brings about the result desired by the user. That interpretation would imply that a letter written with word processing software is a computer program. Whether we should stretch the literal reading of the statutory definition so far is a policy question that goes beyond the scope of this Article. (Because much user output is independently protected as literary, graphic, or other traditional types of works, the question is likely to arise, if at all, only in unusual circumstances.) It should be clear, however, that the specific certain result achieved by the user is not the certain result achieved by the author of the
example, could only do so by rewriting the code to bring about that result.\(^{64}\) Few seem to have realized the profound implications this statutory definition has for the copyright protection of program interfaces.\(^{65}\) It

---

64. See generally Reverse Engineering and Professor Miller, supra note 1, at 990-91.

65. Professor Patry, for example, argues that the Lotus 1-2-3 menu command hierarchy—and therefore presumably any aspect of any user interface generated by program code for application by the user—"functions as a computer program, that is, as statements or instructions that are a subset of the larger 1-2-3 'set' and which themselves bring about a 'certain result.'" Patry, supra note 54, at 17. Notwithstanding his initial insistence that analysis should begin with the statutory definition of a computer program, he does not attempt to distinguish the commands available to the user via the menu command hierarchy from the "statements or instructions" that constitute the code. Nor does he specify what certain result is achieved by the menu command hierarchy. He argues that commands from the menu hierarchy act as programs when they are combined into macros by users, in which case he does define the certain result as "the execution of the user's macros." Patry, supra note 54, at 18. I do not quarrel with his characterization of user macros as computer programs (and although I might have policy reservations about the need to protect individual user macros under copyright, I am unaware that anyone is actively asserting such rights). The author of such a macro, however, is not the Lotus company but rather the individual user of the 1-2-3 program who creates the macro. See supra note 63. That a user-created macro is a computer program argues strongly against the notion that the entire menu command hierarchy is a program in and of itself. The hierarchy, rather, is the instruction set that, together with the rules of combination set by the 1-2-3 program, is available to the user to write macro programs. In other words, the command set, together with the syntactical rules set by the 1-2-3 program, is a programming language. Whether a programming language can itself be copyright protected is perhaps an issue still open to debate. Professor Patry addresses the issue briefly, but I am not clear just what conclusion he reaches. See Patry, supra note 54, at 15-16. In any event, a programming language is not a program any more than the English language is Hamlet. A programming language is a tool or method with which programs—sets of statements or instructions that bring about a certain result—are made. Under Professor Patry's analysis, the instruction set of, say, Intel's Pentium chip would be a protected computer program. That would mean that no other company could attempt to build a competing compatible chip, even if they did so in full compliance with the Chip Protection Act. See 17 U.S.C. §§ 901-914 (1994).

Professor Patry reiterates the same error later in his article. He argues that the Lotus 1-2-3 menu command hierarchy is a set of commands that is independently created pursuant to the exercise of judgment and creativity. See Patry, supra note 54, at 60. I quarrel with none of this. In treating this creative work as a protected computer program, however, he completely fails to specify any certain result that is brought about by the menu command hierarchy, as such, operating directly or indirectly in a computer. A much more natural reading of the statutory definition is that the Lotus 1-2-3 code brings about the certain result that the user is presented with a specific set of commands and a set of rules for combining those commands to accomplish individual tasks determined by the user. Similarly, the protected mask works used to build a microprocessor bring about the result that the chip's physical structure presents the user with a machine language instruction set. That instruction set is used to write programs, such as operating software, but neither the instruction set nor the chip nor the mask works are themselves computer programs. The Lotus menu command hierarchy, in other words, is simply part of that certain result brought about by execution of the Lotus 1-2-3 code. So, Professor Patry's subtitle, "It's All in the Definition," is correct. See supra note 54. He just ignores a crucial part of that definition, namely, the requirement that the set of statements or instructions constituting a protected computer program must be used to bring about a certain result.
means that interfaces in general, and user interfaces in particular, must find copyright protection as traditional classes of works. As is discussed in more detail below, this rules out copyright protection for essentially all nonuser interfaces as falling outside the scope of copyright subject matter. Moreover, although some aspects of user interfaces, such as screen displays, might qualify as pictorial, graphic, or literary works (or as compilations), the traditional limitations on the scope of copyright protection, including § 102(b) and the functionality limitations of Baker v. Selden, apply in full force. The scope of such protection is not affected by the 1980 amendments to the Copyright Act recognizing the protection of functional computer programs—sets of statements or instructions—under copyright.66

We now turn to judicial analysis of the three problems that are the subject of most heated debate in applying copyright law to the protection of computer software. The first problem addresses the scope of copyright protection in a computer program—how much of a

66. Even if we were to approach the issue of copyright protection for interfaces as a part of the copyright in the underlying program, it is clear that what a computer program does—the screen displays it generates and the methodology it presents to the outside world for using the program via such means as keyboards and data entry formats—lies at a higher level of abstraction above literal code than even program structure and organization, whatever level of detailed program structure that courts may end up concluding is covered by a program copyright. Therefore, even under the traditional “abstractions” analysis for determining the scope of protection in a program, user interfaces are relegated to protection (if at all) as other types of copyright subject matter or under other branches of intellectual property law. This analysis, too, leaves § 102(b) and Baker v. Selden as authoritative, notwithstanding that both predate the congressional mandate to protect functional computer programs under copyright.

Professor Patry's primary argument that the menu command hierarchy in Lotus v. Borland constitutes copyright subject matter is that the set of commands, and of themselves, are a computer program. See supra note 65. However, he also argues that the menu command hierarchy is “an integral part of the larger 1-2-3 computer program.” See Patry, supra note 54, at 17. Professor Patry does not attempt to deal with the traditional “abstractions” test in making this argument. Rather, he simply asserts that, because the command hierarchy is not marketed separately and has no existence separate from the program, it should “be analyzed under principles applicable to computer programs.” Id. His premises are not factually correct, in that the menu command hierarchy could be fully generated by an independently coded program, which, in fact, is exactly what Borland did. In that sense, the menu command hierarchy does have an existence separate from the program (code) generating it. It is true that the menu command hierarchy, and indeed all interface elements and all other results brought about by computer program execution, remain abstractions until implemented by code. But it stretches the ordinary meaning of language severely to say that what a computer program does upon execution is a part of the program, as opposed to the result achieved by the program. Moreover, if Lotus were afforded the rights in the menu command hierarchy that Professor Patry believes follow from copyright law, there is every reason to expect that it, or other protected interface designs, would be separately marketable. Interface rightholders might well find it profitable to license rights to generate an identical interface without licensing the underlying code. Then, the competitor would have to do what Borland did, namely, make a substantial investment in coding, debugging, testing, and marketing. That functional interface designers might desire such rights, of course, is no reason to recognize them under copyright, even if they are “creative.” Patent law is the appropriate branch of intellectual property law for the protection of function. See supra notes 2-38 and accompanying text.
program beyond literal code, if anything, should be protected? The second problem deals with the protection of interfaces, including user interfaces. The third problem involves the legality of reverse engineering. The judicial approaches will be compared and evaluated in terms of the more coherent underlying theory of the copyright protection of computer software presented herein.

IV. THE SCOPE OF PROTECTION IN A COMPUTER PROGRAM

I take it as given that the vulnerability of program code to cheap, fast, and easy misappropriation calls for legal protection against verbatim and near verbatim copying of code. Essentially, copyright achieves this result automatically through its exclusive reproduction right. Copyright, however, comes with several centuries of nearly exclusive application to nonfunctional literary, artistic, and musical works, and the policy basis for protecting those types of works often calls for protection beyond verbatim copying, while at the same time denying protection for ideas, procedures, and systems described or illustrated in copyright-protected works. The policy basis for protecting functional works is very different from that applicable to traditional copyright-protected works. 67 If we were writing on a clean slate to determine the level of intellectual property protection for computer programs, therefore, we might consider the advantages of protecting noncode elements of programs—such as program languages, functional user interfaces, or clever algorithms—but for a short period of time. 68 Or, we might consider a compulsory licensing scheme that would provide creators of new and popular program features a financial return from later creators who use them as building blocks but would not allow enjoining the use by others altogether. Under the Berne Convention, however, neither a shorter copyright term nor compulsory licensing for literary works is possible. 69 Therefore, only the scope of protection, under the idea-

67. See supra notes 2-38 and accompanying text.
69. Of course, one can reasonably argue that computer programs are simply not covered by the Berne Convention. The French apparently took that view in adopting, prior to the European Community Software Directive, a 25-year period of protection for programs. On the general question of Berne coverage of programs, see William R. Cornish, Computer Program Copyright and the Berne Convention, in HANDBOOK TO EUROPEAN SOFTWARE LAW 183 (M. Lehmann & C. Tapper eds., 1993).
expression distinction, is available generally for drawing the policy balance for this new type of copyright-protected work.\footnote{70}

Yet it seems clear that limitation along at least one of these three dimensions is necessary, at least absent a showing that some dramatic change from our traditional drawing of the social policy balances is called for. It is one thing to protect computer programs narrowly against verbatim and near verbatim copying of code, or even against other takings made possible by digital technology that too severely undercut original investment incentives. That type of protection may require a change from traditional copyright thinking, but at least we understand why we are doing it. To protect functional works broadly for some seventy-five to one-hundred years, without any provision for compulsory licensing that would permit others to build and improve upon the protected work, and without a showing that the contribution even meets the traditional standards for a twenty-year patent, would be a dramatic increase on the protective side of the balance, without any showing (at least as yet) why this particular technology should suddenly get so much more protection than any other.\footnote{71}

\subsection{A. Computer Associates}

The broad scope of protection afforded to computer programs from many United States courts in the 1980s is well known.\footnote{72} Although the Second Circuit's 1992 decision in \textit{Computer Associates v. Altai} is widely known, it continues to rely on traditional copyright terminology and concepts rather than the fundamental antismisappropriation policy on which program protection ultimately must be grounded, it also supplies a new framework for analysis that, when properly applied, significantly cuts back on the free-wheeling approach of \textit{Whelan}.

The trial court in \textit{Computer Associates} found that there was no similarity between defendant's code (as ultimately rewritten) and plaintiff's code.\footnote{74}

\footnote{70} As important as fair use is in tailoring the appropriate degree of protection under copyright, it remains a highly fact-specific enterprise.

\footnote{71} \textit{See New Protectionism, supra} note 1.


\footnote{73} 982 F.2d 693 (2d Cir. 1992).

\footnote{74} \textit{See id.} at 702, 714.
Therefore, the key issue in the case became what, if any, nonliteral elements of programs were protected. As is now well known, the Second Circuit adopted the "levels of abstraction" approach for determining the protectability of so-called nonliteral elements of a computer program. This approach was purportedly based on the traditional abstractions test for distinguishing idea from expression in literary works, but coupled with "filtering out" of unprotected elements at multiple levels. The court must first separate program into various levels of abstraction, with code the most concrete level, and ultimate function the most abstract level. At each level of abstraction, we must filter out unprotected elements, such as those determined by efficiency considerations, dictated by external factors, or taken from the public domain. Whatever is left is the "golden nugget" of protected expression, which, to the extent it is copied by the defendant, must be further assessed for its importance to the plaintiff's overall program.  

It is important to note that Computer Associates begins by flatly stating that literal object and source code is protected under copyright. This may be the most important part of the decision, because it means that the protection of literal code is not subject to the filtering analysis.

75. Id. at 706-10. One should note that looking at multiple levels of abstraction is itself an extension of, and in important ways unrelated to, the purpose of traditional abstractions analysis, which was to separate idea from expression in a unitary whole. Under the traditional rule, only one abstraction line was drawn. Below that line lay protected expression and above it lay unprotected idea. Everybody agreed that no one knew where even that single line was, as the creator of the "levels of abstraction test," Judge Learned Hand, frankly admitted in devising the "test." See Nichols v. Universal Pictures Corp., 45 F.2d 119, 121 (2d Cir. 1930) ("Nobody has ever been able to fix that boundary, and nobody ever can."). See also Peter Pan Fabrics, Inc. v. Martin Weiner Corp., 274 F.2d 487, 489 (2d Cir. 1960) (Judge Hand stating that idea-expression decisions must necessarily be ad hoc). The Second Circuit's approach in Computer Associates requires us to look for several such lines.

The Computer Associates approach to levels of abstraction essentially means dividing the program up into manageable parts—parts defined by technological considerations and not by lawyers—by retracing the designer's steps. 982 F.2d at 707. The court never explained why the higher levels of abstraction above code are protected by copyright law at all, nor did it try to find an abstractions line analogous to detailed plot sequence in a play above which everything is treated as unprotected idea. By allowing courts—even if only in principle—to find expression at structural levels above literal code, the Computer Associates analysis seems to disregard the definition of a computer program as a set of instructions. See supra notes 53-66 and accompanying text. A much simpler approach that is better aligned with traditional law would be to draw only one abstractions line, as envisioned by Judge Hand in fashioning the abstractions test, somewhere around the literal code and near-verbatim-paraphrase level. Of course, even under this single-line approach, code "expression" lying below the line may not be protected if its reproduction is necessary for compatibility, is in some sense uniquely efficient, or fails traditional copyright tests like merger or scenes a faire. It would make sense, however, to place the burden of proving nonprotectibility of code lying below this line on the defendant who has claimed a need to copy it. That would force the defendant to present plausible reasons for such takings, which alone goes a long way in prohibiting slavish copying—the ultimate goal of the exercise. See New Protectionism, supra note 1, at 87-88.

76. See 982 F.2d at 702. "It is now well settled that the literal elements of computer programs, i.e., their source and object codes, are the subject of copyright protection." Id.
applicable to nonliteral elements. As discussed above, this is a necessary, indeed the crucial, antimusappropriation interpretation that accomplishes the social policy objective of protecting programs from piracy, because if we filtered code for functionality or nonprotectability as a § 102(b) process, we could easily set at naught the congressional decision to protect computer programs under copyright. The importance of leaving code out of the filtering process is underscored when one considers the unlikelihood that any nonliteral elements associated with programs, unless otherwise protected as independent works, such as video game characters or design documentation, will survive the Computer Associates filters.

That few nonliteral elements of value can survive the filtering process follows directly from an honest reading of Computer Associates. The opinion suggests that some "golden nugget" of expression might survive and even that some "quantitatively small misappropriation" may be a "qualitatively vital aspect of . . . protectable expression." Yet, after nonliteral elements related to efficiency, compatibility, and functionality have been filtered out, it is difficult to see how anything remaining could even be important, let alone vital. All programs are written to accomplish a wholly functional purpose. Any esthetic component they may contain, such as a clever new way of accomplishing a particular function with fewer lines of code that a computer scientist might describe as more elegant, is an integral part of

---

77. See supra text following note 52.
78. 982 F.2d at 710.
79. Id. at 714.
80. Professor Patry objects to Computer Associates's use of an efficiency filter on the ground that it is a technical rather than a legal concept. See Patry, supra note 54, at 54-55. He argues that small improvements in efficiency resulting from incremental improvements should get "thin" copyright protection while more substantial improvements should have more extensive protection, reflecting their greater creativity. He does not say how, if a given program element is protected, an incremental improvement avoids infringement under copyright's substantial similarity test. See supra text accompanying notes 22-24. Nor does he explain why works of technology—the only works for which the concept of efficiency has any meaning—that show great creativity need copyright protection, in view of patent law's expressly available protection for nonobvious technological advances.
81. The end product of a program, of course, may simply be a picture or a story, neither of which is functional in the sense used herein because their usefulness inheres solely in their appeal to human beings through their conveyance of information or portrayal of an appearance. But the computer program is the functional means by which this nonfunctional end product is created. A paint brush is functional, even when it is used solely to create a nonfunctional, copyright-protected painting. That these works can now be created by writing computer code instead of using other tools does not make the code itself nonfunctional. We see again the importance of distinguishing between a computer program as a set of statements or instructions and the certain result brought about by machine execution of those instructions. See supra notes 53-66 and accompanying text.
their functionality. Consequentially, as a practical matter, the end result of the Computer Associates analysis, to the extent it is actually followed, may be that only code is protected, and only against literal copying or close paraphrasing. However, the failure of the court to be explicit in this regard has left most later courts still floundering, even where they purport to follow the Computer Associates analysis.

The basic question that Computer Associates failed to address is how any nonliteral program aspect that somehow survives the filtering procedure can be deemed expressive. Every nonliteral aspect of a program, as well as every literal aspect for that matter, exists for the functional purpose of causing the program to perform in some optimal manner in relation to the constraints set by the environment in which it is to be used (including the nature of its intended or expected users). Many courts and commentators seem to think that this is the place for the merger doctrine to come into play—if there is a wide variety of different ways of achieving optimal program performance, or if different programmers working independently of one another would likely arrive at somewhat different solutions, any given solution should be deemed expression.

The difficulty with the merger approach to idea-expression is that Baker v. Selden and § 102(b) of the Copyright Act mean much more than the merger doctrine. The systems and processes described in a copyright-protected work are unprotected no matter how many other systems or processes can be found that accomplish the same result and regardless of whether they accomplish that result less, equally, or more efficiently. The Court in Baker did not inquire into whether other

82. Of course, a nonfunctional trap deliberately placed in plaintiff's program to detect copying would tend to prove slavish copying if also found in defendant's program, but it would hardly be vital to the functionality of the program or to plaintiff's expression. Moreover, a trap is likely to show up only in those cases in which slavish copying of literal code has occurred, and in that case the infringement analysis need not reach the issue of whether nonliteral elements are protected.

83. In the actual case, the district court found no nonliteral element protected, and the Second Circuit found no error in the analysis. See 982 F.2d at 714-15.

84. This, for example, was the approach of Whelan, which the Tenth Circuit in Gates Rubber approved after cutting back on Whelan's overly narrow conception of "idea." Gates Rubber Co. v. Bando Chem. Indus., Ltd., 9 F.3d 823, 840 & n.17 (10th Cir. 1993); see also Note, Steven R. Englund, Idea, Process, or Protected Expression?: Determining the Scope of Copyright Protection of the Structure of Computer Programs, 88 MICH. L. REV. 866, 903 (1990) (arguing for protection if the same process can be implemented efficiently by combining and redividing module functions in a number of substantially different ways).

85. See, e.g., Affiliated Enters., Inc. v. Gruber, 86 F.2d 958 (1st Cir. 1936) (promotional scheme protectable only by patent, regardless of how good it is or how costly it was to develop); Affiliated Enters., Inc. v. Gantz, 86 F.2d 597 (10th Cir. 1936) (similar promotional scheme); Brief English Sys., Inc. v. Owen, 48 F.2d 555 (2d Cir. 1931) (only patent is available to protect a new system of shorthand); Arica Inst., Inc. v. Palmer, 761 F. Supp. 1056 (S.D.N.Y. 1991) (method of describing psychological traits not protected); Kepner-Tregoe, Inc. v. Carabio, 203 U.S.P.Q. 124, 130 (E.D. Mich. 1979) (finding that Kepner-Tregoe had no monopoly on pedagogical technique involved in the management training program); see also Samuelson, supra note 29, at 226-27 & n.73.
accounting methods existed to accomplish the goals of Selden's system (in principle there are millions) let alone any that were better than the one at issue. The fundamental notion is that functional or § 102(b)-type works not meeting the stringent requirements for a twenty-year patent must be allowed to develop through incremental change, via the contributions of many creative persons and not just the person who first arrives at a particular stage.

Courts searching for expression in nonliteral program elements should therefore understand that they are engaging in a metaphysical exercise that deviates fundamentally from copyright tradition. Some people undoubtedly think that such a policy change is desirable, and possibly Congress will someday issue such a directive through legislation. In fact, however, Congress has not yet spoken on the scope-of-protection question for programs—other than to imply that program code, at least, is protected by a program copyright—so a court should require the proponents of radical change to demonstrate the clear desirability of the change as a matter of social policy or that there is no other way to interpret the statute and still give meaning to the congressional directive to include programs under copyright.

Both the definition of a computer program as a set of statements or instructions to be used in a computer to bring about a certain result, and the apparent concern of CONTU with piracy of literal program elements suggest an interpretation that gives substantive meaning to the congressional directive to protect programs under copyright and at the same time minimizes the degree to which traditional copyright must be bent out of shape to accommodate them. This conservative approach is to protect program code under copyright against literal copying and slavish transformations or translations, whether direct or indirect, by modern digital means. This can be achieved by drawing the abstractions line between idea and expression at or near the level of literal code, analogous to the thin copyright protection that is afforded to histories, biographies, rule books, legal forms, and scientific works.

In Russell v. Northeastern Publishing Co., 7 F. Supp. 571 (D. Mass. 1934), the plaintiff had written a book on auction bridge play and charged that defendant had copied a hand therefrom for its newspaper column on bridge (with very minor changes in cards irrelevant to the key point of the instruction). The court found it "clear" that plaintiff had no exclusive right "in the particular distribution of the fifty-two cards," even though there are billions of such card combinations. Id. at 572. The reason, of course, is that the point of instruction is not protected, and the particular hand was a convenient way of making it, although surely not the only one.

86. See 17 U.S.C. § 101 (1994) (definition of "computer program"); see also supra notes 53-66 and accompanying text.
87. See supra note 46.
88. See, e.g., Harper House, Inc. v. Thomas Nelson, Inc., 889 F.2d 197 (9th Cir. 1989) (utilitarian aspects of "organizers"); Landsberg v. Scrabble Crossword Game Players, Inc., 736 F.2d 485 (9th Cir.
This would simply end the debate over whether and how nonliteral program elements can be expressive, until someone demonstrates a plausible ground for distinguishing program design technology from other technological designs for the purpose of bringing it under copyright instead of patent and trade secret. Once such a ground is demonstrated, we will then have a standard for deciding the otherwise metaphysical idea-expression question for computer programs. Substantial side benefits, of course, would be much higher predictability, reduced costs of litigation, and more efficient use of scarce judicial resources.

B. Subsequent Decisions on Scope of Program Copyright Protection

Nearly all courts subsequent to Computer Associates purport to accept and follow its reasoning and analytical approach. That many of these cases deal with user interfaces, however, rather than the program generating the interfaces, indicates the degree to which Computer Associates has failed to provide a clear road map on the scope of program copyrights. Indeed, some of the early cases paid lip service to the new approach introduced by Computer Associates while continuing to follow the broad brush and highly protective approach of Whelan. More recent decisions, however, appear to be narrowing the scope of program copyrights significantly. This subsection considers only cases in which

1984) (game strategy); Adler, 650 F.2d at 1371-72 (holding that research is not copyrightable); Hoebling v. Universal City Studios, Inc., 618 F.2d 972, 980 (2d Cir. 1980) (historical theory); Rosemont Enters., Inc. v. Random House, Inc., 366 F.2d 303 (2d Cir. 1966) (biography); Continental Cas. Co. v. Beardsley, 253 F.2d 702 (2d Cir. 1958) (legal form); Egan-Trego, 203 U.S.P.Q. at 130-32 (E.D. Mich. 1979) (outlines for problem solving). See supra notes 55-59 and accompanying text. The courts have found all expression in these works in the literal and near-literal elements. This policy choice prevents the courts from getting bogged down in the otherwise wholly metaphysical question of whether nonliteral structural or theoretical elements that may be needed by later authors attempting to solve a similar problem constitute expression. It also eliminates the need to address the thorny fact issues that a quality standard (how many ways are there to achieve the same result equally well?) necessarily generates.

infringement of some aspect of the actual computer program was alleged. The next section takes up the user interface cases, even those that purport to rely to some extent on Computer Associates.

In CMAX/Cleveland, Inc. v. UCR, Inc., an early post-Computer Associates decision, the defendant used a licensed copy of plaintiff's program—a copy apparently held under at least partially false pretenses—to generate the various screen images produced by the program. A code generator was then used to produce new code that would generate the same images on the screen. This type of near slavish copying—at least without more—seems a good candidate for appropriate legal prohibition. For this reason, the court's finding of infringement probably comports with the social policy balances appropriate to programs.

Nevertheless, the CMAX opinion is replete with references to defendant's copying of plaintiff's "system" and even the "design of the system," together with observations that system design is the most creative and intellectually demanding phase of software development. The infringement analysis, supposedly following the Computer Associates abstractions approach, divides the program design into file structures, transaction codes and screens, and file reports (without ever questioning whether program design is something that can be protected under the Computer Associates analysis or § 102(b)) and concludes that these elements of plaintiff's program were the foundation of the same elements of defendant's, which was therefore an infringing derivative work.

The breadth and abstractness of the CMAX analysis is more reminiscent of Whelan than of Computer Associates. The court assumed without discussion that creative program design is a protected element of a program copyright, it treated selection and arrangement of field definitions within files as expression without consideration of their

91. Such code generators apparently permit the speedy coding of sophisticated programs with "relatively little creative or intellectual effort." Id. at 343. A competing program created by such methods could seriously undercut original program creation incentives while providing little possibility of an improved work or further technological advance.
92. Id. at 343, 344 n.3, 345, 349, 354.
93. See id. at 354-55.
94. See id. at 355. The court's reliance on the statutory definition of a derivative work—a work "based upon" a preexisting work—as a substitute for analysis is particularly unfortunate. Unless analysis independently shows that defendant has incorporated protected elements into his work, the result cannot be an infringing derivative work, no matter how much he otherwise relied on the original. See infra note 164.
possible status as a system, and it protected transaction codes the standardization of which could more easily permit users familiar with one system to switch to the other. Although the result may well be appropriate under a misappropriation analysis, it cannot be said that Computer Associates was effective in eliminating judicial confusion in reaching it. This type of reasoning is likely to continue until the true policy basis for the copyright protection of programs—protection against competitively unfair methods of copying—is explicitly recognized.

An even more questionable purported application of Computer Associates is found in Autoskill, Inc. v. National Educ. Support Systems, Inc., which involved software used for testing, diagnosing, and training students in reading skills. As there was no claim of code infringement, the court relied on the district court’s comparison of what the two programs did and how they did it. In its list of similarities, the court found that both “programs” employed the same unique techniques and the same “keying procedure” evidencing at least minimal creativity, that students in each received continuous reinforcement.

96. See 804 F. Supp. at 355. This is essentially the user interface problem and user “lock-in.” More recently, courts have come to a contrary conclusion. See, e.g., Lotus Dev. Corp. v. Borland Int’l, Inc., 49 F.3d 807 (1st Cir. 1995), aff’d by an equally divided Court, 116 S. Ct. 804 (1996); Mitel, Inc. v. Iqtel, Inc., 896 F. Supp. 1050 (D. Colo. 1995); see infra notes 191-99, 212-16 and accompanying text.
97. All of the broad scope-of-protection analysis of CMAK was unnecessary for a finding of infringement even without adopting the analytical approach suggested herein. The court independently based infringement on breach of the license agreement and on transferring a copy of plaintiff’s program without permission. See CMAK, 804 F. Supp. at 556-57.
98. 994 F.2d 1476 (10th Cir. 1993).
99. See id. at 1491 n.18.
100. Both the trial and appellate courts in Autoskill hopelessly confuse the computer program with the pedagogical reading “program” or system that the computer program was written to implement. Both use the term “program” without any sign of recognition of the very different meanings the term could have on these facts. Had the courts recognized that the case involved primarily, if not exclusively, a claim for protection of teaching methodology, there would have been no need for a Computer Associates analysis. The teaching system would be unprotected under § 102(b). See Kepner-Tregoe, Inc. v. Carabio, 203 U.S.P.Q. 124, 130 (E.D. Mich. 1979) (system for teaching problem-solving techniques not copyright protected). See generally Samuelson, infra note 29, at 226-27.
101. See 994 F.2d at 1494-95. The court made the fundamental mistake of thinking that “creativity” alone was sufficient for copyright protection, without recognizing that § 102(b) precludes protection for elements of protected works such as ideas, systems, and procedures without regard to creativity. Thus, the court emphasizes the distinctions between the “techniques” plaintiff developed from the theories in the published literature and the “considerable investigation and research” that went into developing plaintiff’s unique “system.” Id. at 1495.
102. Id. Because the defendant failed to show that this procedure was a common practice or dictated by efficiency considerations, the court upheld the lower court’s decision not to filter out these elements in the Computer Associates analysis. Id. at 1496. Again, the court failed to recognize that “procedures” are excluded from copyright protection under § 102(b), without regard to creativity or efficiency.
and immediate feedback, both "programs" used alternating sense and nonsense words, the criteria for progressing to the next stage and the presentation of skills hierarchically from the simple to the complex were common to both programs, and minor changes in format were not pedagogically significant. All of these factors relate not to the computer program at all, that is, to the set of statements or instructions, but rather to what the program does, those certain results that computer execution of the instructions brings about. The court should have recognized that it was dealing with the user interface, not the computer program. Moreover, most, if not all, of these similarities relate to factors that a careful Computer Associates analysis should have filtered out as functional or as § 102(b) processes, systems, principles, or ideas. Thus, the Autoskill case, too, although purporting to reject Whelan and follow Computer Associates, is in fact much more closely aligned with Whelan and its progeny.

The federal district court in Gates Rubber Co. v. Bando American, Inc., declined to follow Computer Associates and expressly treated the behavior of a program as an object of copyright protection, finding infringement notwithstanding the absence of any literal copying of code. The programs involved assisted in the selection of replacement belts for industrial manufacturing machines. The court defined the "central

---

103. Without recognizing it, the court indicated the patent-relevant value of these aspects of the system devised by the plaintiff in referring to the testimony of a school superintendent pleased that he had "finally located a software program that was going to impact the reading instructional program in their school system and in their southwest . . . ." Id. at 1495-96.

104. Id. at 1496.

105. See id. at 1497.

106. See id. The court also found similarities in speed-and-accuracy profiles and in the graphs used in both systems for similar purposes. See id. Infringement on this ground may be more solidly based than on the other similarities, in that the similarities appear to involve graphic works.

107. The court was confused beyond repair by the term "program." See supra note 100.

108. The lower court treated similarities in "important pedagogical aspects" of the two systems as outweighing differences in the logic flow between the display screens in its infringement analysis: "From the student or teacher's standpoint the content of what is communicated is more important than how he or she got there." Autoskill, Inc. v. National Educ. Support Sys., Inc., 798 F. Supp. 1557, 1570 (D.N.M. 1992) (emphasis added). But if something serves important pedagogical purposes, it must be filterable under a Computer Associates analysis. Surely copyright cannot give a 75-year monopoly in effective teaching methods. Equally clearly, the content of a protected work expressing an unprotected system for achieving a given task (such as teaching people to read) is precisely what copyright does not protect. Under § 102(b) and Baker v. Selden, such systems and methods can be protected only through patents, regardless of how creative they are.

issue” as concerning the “calculation methods” implemented by the program, without recognizing that § 102(b) might call into question whether calculation methods are appropriate targets of copyright protection. Moreover, the court based its ultimate finding of infringement on “special regard” for the copying of certain mathematical constants developed by the plaintiff for use in its formulas, in direct conflict with the Supreme Court’s decision in *Feist*, which expressly held that facts—such as numbers—are not protected by copyright.

On appeal, the Tenth Circuit in *Gates Rubber* largely redeemed itself from its disastrous opinion in *Autoskill*, expressly adopting and expanding upon the *Computer Associates* analysis. In addition to filtering ideas, the court indicated that processes found as part of the system architecture, operations within modules, or algorithms, must be filtered out under § 102(b). It also expressly reversed the district court on the protectability of the mathematical constants and remanded on the issue of whether the “control flow” and “data flow,” which the lower court found to be protected elements without specifying exactly what it meant by the terms, should not be filtered out as unprotected.

110. *Id.* at 1503, 1515.
111. 17 U.S.C. § 102(b) (1994) provides that copyright in a work does not “extend to any idea, procedure, process, system, method of operation, concept, principle, or discovery . . . .” *Id.* Calculation methods are at least arguably a procedure, process, or system, if not also a method of operation.
112. 798 F. Supp. at 1519, 1522. The court earlier described these numbers as lying “at the heart of the dispute.” *Id.* at 1518.
115. *See id.* at 836-37. The Tenth Circuit conceded that both intermediate “structure, sequence, and organization” and some “algorithms” might be protectable, subject to the merger or industry-standard filters. *Id.* at 836, 845. It supplied no standard, however, for distinguishing “expressive” structure or algorithms from unprotected “idea” or “process” structure or algorithms. Indeed, given the functionality of computer program structure and the goals for which algorithms are employed in computer programs, it is difficult to guess what such a standard might be. Programs are not structured whimsically, nor are algorithms employed for fun or esthetic beauty. Program structures and algorithms are employed, within the limits of the programmer’s skills, to make the program work better in some sense—to run faster, to make better use of available resources, or to be easier to learn, repair, or use, for example. Later programmers who observe the use of a particular structure or algorithm in a program may recognize that it works better than whatever they had been using before, and they may adopt it for that reason, perhaps even improving upon it in the process. That is the way technology advances—through incremental improvements on the existing base. If copyright protects a particular structure or algorithm, not only is exact copying prohibited but also all substantially similar improvements. Consequently, copyright becomes akin to a patent, subject, indeed, to independent discovery of the hypothetically better structure or algorithm but prohibiting anyone from using the technological advance who learns it from the existing base.
116. *See id.* at 842-43.
processes. Like *Computer Associates*, the court did not fully realize the implications of its approach for the protection of nonliteral program elements; it certainly did not expressly limit protection to literal code (or even derivatives of literal code) and even suggested, without supplying any standard, that some aspects of structure may be "expressive." Still, honest application of the principles laid down in the opinion to the reality of computer programs as functional works—together with the given congressional intent to protect at least code—leaves very little besides literal code under copyright protection.

The Eleventh Circuit’s decision in *Bateman v. Mnemonics, Inc.* is one of the more important computer program scope-of-protection cases since *Computer Associates*. In reversing a jury finding of infringement involving alleged copying of both literal and nonliteral elements of a program for automated parking systems, as well as certain interface features, the court emphasized that some sort of filtration to eliminate unprotected elements must occur at both literal and nonliteral levels, notwithstanding that the *Computer Associates* test was formulated to deal with only nonliteral elements. It refused to hold that interface specifications were unprotectable as a matter of law, but did recognize that the jury had to be instructed to consider external requirements of compatibility as negating copying that would otherwise be infringing. The court was concerned, moreover, about protecting functionality under copyright; the opinion states outright at several points that copyright cannot protect functionality or patentable subject matter. *Bateman* is one of the few decisions explicitly to recognize that *Baker v. Selden* and § 102(b) are not disarmed by the absence of merger (the existence of alternatives to accomplish the same result). However, the

117. *Id.* at 844.

118. *See supra* note 115.

119. *See supra* notes 39-47 and accompanying text. The Tenth Circuit in *Gates Rubber*, like *Computer Associates*, expressly held that literal code is protectable, subject to the merger and *scènes à faire* doctrines. *9 F.3d* at 836.

120. *79 F.3d* 1532 (11th Cir. 1996).

121. *See id.* at 1545.

122. *See id.* at 1546-47.

123. For example:

It is particularly important to exclude methods of operation and processes from the scope of copyright in computer programs because much of the contents of computer programs is patentable. Were we to permit an author to claim copyright protection for those elements of the work that should be the province of patent law, we would be undermining the competitive principles that are fundamental to the patent system.

*Id.* at 1541 n.21 (citation omitted). "In no case... should copyright protection be extended to functional results obtained when program instructions are executed and such results are processes of the type better left to patent and trade secret protection." *Id.* at 1547 n.33.

124. The court stated:
court confessed to finding tension in determining what "process" or "method of operation" actually mean in the context of computer programs.\textsuperscript{125}

The Bateman court was clearly hesitant to state boldly that computer programs are a unique kind of literary work and to draw the conclusions that follow from applying traditional principles conservatively to this new type of subject matter. Its problem stemmed from the failure to recognize the possibility that one can distinguish functional computer program code—based on the statutory definition of a computer program and the policy analysis given above\textsuperscript{126}—from other aspects of program functionality. The court was correct in saying that some sort of filtering process must take place at both levels, but the congressional decision to place functional program code under copyright protection necessarily means that the literal elements require filters different from those applied to the nonliteral elements. In particular, it is the functionality filter that must be different. Functionality, as such, cannot be a bar to protection of code, or else no programs would be protected (because all code is functional), and although some sort of merger filter is necessary,\textsuperscript{127} it remains a difficult question what kinds of efficiency at the code level should be protected in the absence of merger.\textsuperscript{128} There

\begin{itemize}
\item The availability of alternatives should not be determinative in distinguishing between elements of a computer program that are expressive and those that are unprotectable under 17 U.S.C. § 102(b). Generally, there is more than one method of operation or process that can be used to perform a particular computer program function. However, methods of operation and processes are not copyrightable.
\item \textit{Id.} at 1546 n.29; \textit{see also supra} notes 84-85 and accompanying text.
\item 79 F.3d 1546 n.29.
\item See supra notes 39-66 and accompanying text.
\item Recall the language of CONTU:
\begin{quote}
[C]opyright protection for programs does not threaten to block the use of ideas or program language previously developed by others when that use is necessary to achieve a certain result. When other language is available, programmers are free to read copyrighted programs and use the ideas embodied in them in preparing their own works . . .
\end{quote}
\end{itemize}

CONTU REPORT, supra note 45, at 20 (first emphasis added).

128. For example, if someone has discovered a highly efficient block of code, is it protected from copying if there are other ways to achieve the same functional result, even if those other ways are somewhat less efficient? Much less efficient? Does the block of code lose copyright protection if it can be shown that it is in some sense uniquely efficient? Can a second programmer take the efficient block as long as she invests roughly equivalent time and effort in developing the rest of her program independently? Trying to separate idea from expression based on efficiency considerations raises some fundamental issues, which I have long suggested be resolved not by metaphysics but by antimisappropriation reasoning. \textit{See New Providence, supra} note 1, at 83, 87-88; \textit{see also Peter S. Menell, An Analysis of the Scope of Copyright Protection for Application Programs, 41 Stan. L. Rev. 1045, 1085-87 (1989)} (analyzing efficiency considerations in the protection of program structure). The problem is analogous to what we now face with copyright protection for buildings. Congress has removed "architectural works" from the separability test applicable to PGS works, but the degree to which \textit{Baker v. Selden} and other traditional functionality limitations still apply was left open. \textit{See supra} notes 35-56 and accompanying text. In 1989 a group of copyright law professors, who
is no evidence, however, that Congress intended copyright to trench further into patent subject matter beyond code, and until Congress does speak to the question, all of the traditional functionality limitations on copyright protection should apply to the nonliteral elements. These problems are likely to remain until courts learn to separate computer programs as sets of statements or instructions from the certain results brought about by machine execution of those instructions. It is not that the problems then become easy or self-answering, but they are then at least conceptually tractable.

The Second Circuit's recent decision in *Softel, Inc. v. Dragon Medical & Scientific Communications, Inc.*\(^{129}\) rather dramatically shows the difficulty courts have in keeping fundamental principles in mind trying to apply the *Altai* filters. At issue were programs useful in preparing films and videotapes conveying information about complex medical phenomena.\(^{130}\) The trial court had denied infringement of certain programs produced by the defendant after the commencement of the litigation that were written in different computer languages and designed to work with different hardware than plaintiff's programs.\(^{131}\) The Second Circuit accepted plaintiff's argument on appeal that the lower court should not only have considered the individual elements of plaintiff's program, which were perhaps unprotectable in themselves,

---
\(^{129}\) *LaST Frontier Conference on Copyright Protection of Computer Software*, 30 JURIMETRICS J. 1, 19 (1989). They did not expressly apply this reasoning to literal code, which they all agreed was protected, but their concurrent statement that § 102(b) limits code protection indicates their view that functionality or efficiency must play some role in separating protected from unprotected elements. *See id.* at 18.

\(^{130}\) 118 F.3d 955 (2d Cir. 1997).

\(^{131}\) *See id.* at 960.

*See id.* The trial court had found infringement with respect to the programs produced by the defendant that had given rise to the litigation. Given that the programming languages and hardware were different for the postlitigation programs, it is difficult to see how the defendant escaped doing roughly the same amount of work in developing them as the plaintiff. If this is true, it would mean that the antispecies appropriation rationale for protecting software under copyright would not demand a finding of copyright infringement. *See New Protectionism,* supra note 1, at 88-89. In this case, of course, the defendant might also have shortened plaintiff's lead time by appropriating trade secrets contained in the plaintiff's programs. Plaintiff had hired by defendant specifically to develop the programs at issue, and plaintiff had delivered only executable forms of the programs, initially, at least, without apparent objection by the defendant. *See id.* at 958-59. The trial court found liability for the theft of some trade secrets, and the Second Circuit remanded for consideration of whether the program structure, too, was a trade secret. *See id.* at 968-69. The existence of trade secret claims, however, should not affect the copyright analysis.
but also whether those elements were combined in an expressive way and whether defendant had copied that expression.\textsuperscript{132} The opinion reiterates the possibility of finding copyright-protectable expression at the higher levels of program abstraction,\textsuperscript{133} and in doing so, it completely fails to recognize how the fundamental functional nature of computer programs distinguishes them from traditional copyright subject matter.

For example, \textit{Softel} analogizes computer program design to an abstract painting made up wholly of individually unprotected geometric figures or a literary work made up of individually unprotected words, where in both cases the combination is original and expressive.\textsuperscript{134} For traditional works such as paintings and novels, of course, the original combination of unprotected basic elements is precisely what we have always called expression for purposes of copyright protection. However, an absence of nearly verbatim copying leads to a finding of noninfringement of even traditional copyright works, such as histories, legal forms, and technical works, where the copyright is thin.\textsuperscript{135} The beguiling but wholly inapt analogy to the esthetic core of copyright, such as paintings and novels, causes courts like that in \textit{Softel} to lose sight of the basic policy goals for protecting technological works like computer programs under copyright law and runs the risk that they will ultimately obliterate the boundary between patent and copyright.

The \textit{Softel} opinion also appears to make the mistake of equating expression at structural program levels with the mere absence of merger. For example, it quotes with apparent approval the plaintiff's argument that there were other ways to accomplish the functions performed by the plaintiff's program.\textsuperscript{136} The court may have been on firmer ground in indicating that specific commands were copied by the defendant,\textsuperscript{137} but even here the court failed to relate these commands to the definition of a computer program as a set of statements or instructions. Rather, it seems that for the \textit{Softel} court, merger was the sole test of the existence of expression in the structure of a computer program.\textsuperscript{138}

\begin{enumerate}
\item[132.] \textit{See id} at 963-64.
\item[133.] \textit{See id. at} 963.
\item[135.] \textit{See supra} notes 57-58 and accompanying text.
\item[136.] \textit{See Softel,} 118 F.3d at 965.
\item[137.] \textit{See id. at} 966.
\item[138.] The court states the basic issue as "how many ways existed to design a computer program with the same functionality as Softel's." \textit{Id.} Near identity of the use of specific commands indicated to the court that plaintiff had "at least a colorable claim" that its design included expression. \textit{Id.}
\end{enumerate}
At its most extreme, Softel might therefore be read to represent nearly a complete retreat from the implications of Altai back to the pure merger test of Whelan. However, the ground for reversal was narrower than much of the language of the opinion. In particular, the Second Circuit found that the trial court had erroneously failed even to consider the application of the Altai filters to the program design and sent the case back for a determination of whether the combination of design elements was protectable expression.\textsuperscript{139} Therefore, the lower court should still be bound by Altai’s requirement to filter out functional elements related to efficiency in addition to combination elements that are denied protection under \textit{scenes a faire}, merger, or other limiting doctrines. Properly effected, this analysis is unlikely to find any expression at the level of program structure, because all programs are designed for the purpose of running faster, being easier to use or repair, and generally better accomplishing their intended function with the available resources. Nevertheless, this particular Second Circuit panel has demonstrated a profound myopia in treating computer programs as copyright subject matter similar to novels and paintings. It has not helped the cause of advancing understanding and eliminating confusion concerning the copyright protection of computer software. It may even have set the cause back several steps.

A few other post-\textit{Computer Associates} decisions are worth mentioning briefly. One decision, \textit{Harbor Software, Inc. v. Applied Systems, Inc.},\textsuperscript{140} analyzes several program structure and interface aspects and, basically, finds copyright protection for all of those for which alternative choices were available, including the selection and arrangement of data categories. This overly simple analysis fails to recognize that § 102(b) and \textit{Baker v. Selden} go well beyond merger to the heart of the distinction between patent and copyright subject matter.\textsuperscript{141} A later decision limits protection in some of these elements, however, on the ground that they showed only a “minimum of creativity” and consequently would not be infringed when found in a work that differs in “more than a trivial degree.”\textsuperscript{142} Another district court decision holds that a request for bids on a computer software system might infringe the copyright in a computer program on which it was allegedly based through its adoption of protected nonliteral elements, even where access to source code was

\begin{footnotesize}
\begin{enumerate}
\item[139.] See id. at 967.
\item[140.] 925 F. Supp. 1042, 1049-52 (S.D.N.Y. 1996).
\item[141.] See supra notes 84-85, 124 and accompanying text.
\end{enumerate}
\end{footnotesize}
not alleged.\textsuperscript{143} Even this decision, however, is careful to point out that nonliter elements necessarily incidental to program function, including those dictated by considerations of efficiency, are not protected.\textsuperscript{144}

An interesting case is \textit{Baystate Technologies, Inc. v. Bentley Systems, Inc.}\textsuperscript{145} Here the court was faced with the problem of competing programs conceded to be using data structures that were somewhat similar. The program in question was a translator that would allow plaintiff's computer aided design (CAD) application program, named CADKEY, to work with another CAD program developed by defendants. The court found copying of the structure as a factual matter,\textsuperscript{146} but concluded that \textit{Computer Associates} precluded copyright protection. External factors governing the organization and selection of the data files were found to be the need for compatibility and industry-wide programming practices. The defendant's translator program could not function properly without reading the data files of CADKEY, which in turn, required that the translator program be organized in a similar manner.\textsuperscript{147} Perhaps even more important, the court recognized that defendant should not have to choose different labels for internal program variables just to be different, when programming efficiency, as a matter of common sense, made use of the same labels a reasonable and normal programming choice; the plaintiff's own materials made it clear that compliance with its internal structures was necessary to avoid chaos.\textsuperscript{148} Moreover, a neutral industry expert testified that, in developing translators, it was his practice to use, at least to some extent, the names and organization of the target product's data structures.\textsuperscript{149}

In sum, courts are slowing coming to realize that healthy competition in the markets for computer software does not require wholesale

\textsuperscript{144} See id. at 49.
\textsuperscript{146} See id. at 1086-87.
\textsuperscript{147} See id. at 1088-89.
\textsuperscript{148} See id.

\textsuperscript{149} The court also concluded that, even if the data structures were not filtered out as copyright-protected elements, there was no infringement in this case because they were "neither a substantial portion nor a significant aspect of the whole copyrighted work." Id. at 1089.
abandonment of our traditional distinctions between patent and copyright subject matter in the digital age. Except possibly for Softel, the protective high-water mark of Whelan would seem now safely in the past. Fully coherent judicial discussion of the policy basis for their decisions must await conscious recognition of a theory of software protection that is consistent with the statutory definition of a computer program as a set of statements or instructions distinct from the certain result that those instructions bring about upon execution in a computer. To the extent courts honestly apply the Computer Associates filters, they will eventually reach the point at which code is the only element of programs that is protected, and only against slavish or near verbatim copying or translation, by the program copyright. Even to the extent that courts continue, incorrectly, to apply Computer Associates to interfaces, they are still likely to end up limiting the copyright protection of function, which is to say they will come to the correct result if for the wrong reason (as indicated by the Bateman court's application of Computer Associates to interface features). The next section further develops the correct approach to interface protection under copyright and compares this coherent theory to current judicial approaches.

V. INTERFACE PROTECTION UNDER COPYRIGHT

The copyright protection of computer program interfaces, and user interfaces in particular, requires special attention. Nearly all courts dealing with the problem to date, including courts that reach the same conclusions as those advanced by the theory of software protection advanced herein, have treated interfaces simply as a nonliteral element of the computer program generating or implementing the interface. This approach assumes away the fundamental question of whether a particular interface constitutes copyright subject matter. As the analysis in this section shows, many interfaces are not, in fact, copyright subject matter, and courts should be aware of the need to dismiss copyright claims with respect to such interfaces. Moreover, even where aspects of a user interface can be considered copyright subject matter—as a graphic, pictorial, or literary work, for example—its copyright status is independent of the copyright in the computer program that generates it. Consequently, all of the traditional limitations on copyright

---

150. This section is based heavily on Lotus Amicus Brief, supra note 2. See also Dennis S. Karjala & Peter S. Menell, Applying Fundamental Copyright Principles to Lotus Development Corp. v. Borland International, Inc., 10 HIGH TECH. L.J. 177 (1995) (publication in article form of an amicus brief filed with the First Circuit in the Lotus case); Dennis S. Karjala, Interfaces, Remarks at the Second International Symposium on Legal Protection of Computer Software (Nov. 7-8, 1989) (transcript available from author).
protection apply, including the functionality limitations of *Baker v. Selden* and § 102(b) of the statute.

As discussed above, under the Copyright Act, 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."\(^{151}\) Unless the certain result effected by a program is independently protected under copyright law, anyone has the right to produce independent code bringing about even an identical result. Software interfaces, including user interfaces, are an important part of that certain result that the set of instructions accomplishes.

This straightforward conclusion—that what a program *does*, and therefore its appearance to the outside world when viewed solely through input and output devices, is not protected by the copyright in the program—was explicitly recognized by the Second Circuit in *Computer Associates*:

> [W]e note that our decision here does not control infringement actions regarding categorically distinct works, such as certain types of screen displays. These items represent products of computer programs, rather than the programs themselves, and fall under the copyright rubric of audiovisual works.\(^{152}\)

Therefore, the analysis of *Computer Associates* and other cases dealing with the scope of copyright protection for nonliteral elements of program code simply does not address the central issue of interface protection, which involves the protection of computer program outputs. The scope of such protection is circumscribed by all of copyright's traditional limiting doctrines and is not affected by the 1980 amendments to the Copyright Act recognizing the protection of computer programs—sets of statements or instructions—under copyright.\(^{153}\)

\(^{151}\) See *supra* notes 53-66 and accompanying text.

\(^{152}\) 982 F.2d at 703. For a more detailed explication of the distinction between a computer program and its user interface, including some of the early judicial and administrative errors in addressing this issue, see Samuelson, *supra* note 29, app. at 264-69.

\(^{153}\) Even under the traditional abstractions test, the program interfaces must be excluded from protection because they lie at a higher level of abstraction than even the hotly disputed area of structure, sequence, and organization. See *supra* note 66. This highlights, however, one of the fundamental conceptual difficulties of the *Computer Associates* approach to abstraction. By allowing the possibility of finding expression at multiple levels of abstraction, the *Computer Associates* approach not only deviates fundamentally from the traditional abstractions test, see *supra* note 75; it also *automatically* brings within the range of potential copyright subject matter everything associated in any way with a program, without consideration of its nature as a set of statements or instructions or any other class of copyright subject matter.
A. Interfaces as Copyright Subject Matter

Where do interfaces come from? Interfaces allow interoperability between hardware and software (hardware-software interfaces), between different software programs (software-software interfaces), and between software and human users (user interfaces). Although all such interfaces are generated—or, some might prefer to say, implemented—by computer programs, usually in conjunction with other programs (like operating software) and the hardware on which the programs run,\(^{154}\) the interfaces themselves are not programs. In fact, although any particular use of an interface will involve concrete physical signals—inputs from a keyboard, outputs on a screen, or binary signals between programs, for example—the complete interface itself is really no more than the abstract set of rules or definitions that specify how the program generating the interface responds to input signals or provides output signals. The interface may often be designed before any code is written to implement it, but the interface itself does not and can not exist before implementation programs are written, and even then there is nothing in the physical world to which one can point and say, "This is the interface."\(^{155}\) Consequently, an interface is the result of a collaborative effort between the interface designer and the program creator. The interface designer may well be the more creative of the two in any given case, but careful legal analysis must recognize a distinction between their separate efforts. We must further bear in mind that creativity alone is not enough for copyright protection. Rather, the creativity must inhere in a "work of authorship."

Because interfaces are the methodology by which particular hardware or hardware-software combinations are put to specific use, patent law would appear appropriate for determining the protection to which they are entitled.\(^{156}\) The availability of copyright protection for interfaces is much more problematic if we are to remain true to the statutory language and bear in mind the true nature of interfaces. Except for literary, graphic, or pictorial aspects of user interfaces, in fact, it is

---

154. Some interfaces may be created directly by the hardware alone, as in the case of the early machine languages, but the analysis of their legal protection should not be different from that for other interfaces except to the extent that the analysis somehow depends on the relationship between an interface and a program that implements it. In my opinion, neither as a matter of policy nor under the current Copyright Act should the express statutory protection for programs affect the protection analysis for interfaces, whether the interfaces are implemented solely through hardware or through programs.

155. There are, of course, hardware devices, such as keyboards, that must be used to transmit signals through the interface, but they do not constitute the interface of any particular program.

difficult to see how copyright protection could attach at all, except as a nonliteral element of the program copyright. But the program definition itself distinguishes the certain results brought about by operation of the program from the statements or instructions that constitute the program. However far protection may extend into the penumbra associated with the set of statements or instructions, such as to their structure or organization, the statute excludes the certain results from the program definition. Because the interfaces are a part of that certain result achieved by the program, they too are excluded from the program copyright. If interfaces are to be protected, therefore, they must somehow qualify independently as copyright subject matter.

Some aspects of user interfaces, such as screen displays or help explanations, may independently qualify as pictorial, graphic, or literary works. The degree of copyright protection in such works must then be determined under traditional copyright law applicable to works of those types, modified perhaps within rational boundaries to fit the nontraditional digital environment. Nonuser interfaces, however, are simply not "writings," nor do they fall within any of the categories directly protected by the Copyright Act.\textsuperscript{157} A program, of course, is a writing that generates the interface (in the sense that the program responds to input-output signals in the manner for which it is programmed), but the interface itself is no more a writing than the method of operation we might learn from a book that describes how to operate a machine. Moreover, certain aspects even of user interfaces have a difficult time qualifying as independent works.

A good example is the menu command hierarchy at issue in \textit{Lotus v. Borland}.\textsuperscript{158} The First Circuit distinguished between nonliteral copying of program code and literal copying of the menu command structure, and on this basis the court found that the \textit{Computer Associates} analysis did not apply. This conclusion was correct, but not because the copying was either literal or nonliteral. The \textit{Computer Associates} analysis did not apply for the reason given in \textit{Computer Associates} itself, namely, that the interface is not covered by the program copyright.\textsuperscript{159} Because of this misunderstanding, the First Circuit did not determine whether the menu command hierarchy falls within the intended scope of copyright subject matter.

A menu command hierarchy, in fact, does not fit comfortably as an independent work within any of the classes of works set forth in \textsection{} 102(a)

\textsuperscript{157} See 17 U.S.C. \textsection{} 102(a) (1994) (defining copyright subject matter as "original works of authorship" and including examples such as literary works, musical works, audiovisual works, etc.).

\textsuperscript{158} 49 F.3d 807, 814-15 (1st Cir. 1995).

\textsuperscript{159} See infra note 152 and accompanying text.
of the Copyright Act (literary works and musical works, for example). Although Congress intended the actual list of categories as "illustrative and not limitative," Congress must have intended § 102(a) as a whole to provide bounds on copyright subject matter. A computer program's menu command hierarchy is not even roughly analogous to any of the § 102(a) categories of copyright subject matter. It is a conceptualized collection of commands for operating a machine that is under the control of a computer program and is most closely analogous to a process that is at the heart of patent subject matter. It is not a collection of preexisting materials or of data so as to constitute a "compilation" within the statutory definition. Neither is it a nonliteral element of the screen displays, any more than Selden's system for using his accounting forms was a nonliteral element of those particular graphic works.

It is true that the menu command hierarchy represents a collection of functions available to the user and the methodology, arguably quite creative, for implementing those functions. However, it cannot be the case that any work manifesting human intellectual creativity is copyright subject matter. Every patented invention, whether machine or process, necessarily manifests human intellectual creativity by reason of patent law's nonobviousness requirement. An automobile, for example, may equally be considered a collection of individual functions and features, yet no one has seriously argued that an automobile, or any particular collection of features offered by a given model, is therefore a copyright-protected compilation.

161. See 35 U.S.C. § 101 (1994). Professor Patry has argued that the menu command hierarchy is itself a "computer program." For a critical analysis of this position, see supra note 65.
162. 17 U.S.C. § 101 (definition of "compilation").
164. Perhaps closer to home would be an argument that a physical microprocessor is copyright protected as a compilation through its offering of a creative selection or arrangement of an instruction set. Cf. supra note 65 (discussing the Lotus menu command hierarchy's status as an independent computer program). Such an interpretation cannot have been Congress's intent, as it would extend copyright protection to every product on the market that could be said to offer a creative combination of features.

Another superficially appealing argument that cannot withstand careful analysis is that Borland created a derivative work in basing Quattro Pro on the Lotus 1-2-3 program, even if Borland did not take any protected elements from the Lotus work. On the assumption, however, that a menu command hierarchy is not copyright subject matter or, if it is, that only unprotected elements were taken from it, writing independent code generating even an identical menu command hierarchy cannot infringe a § 106(2) right to create derivative works. In Lotus, for example, it was true that the Borland emulation mode command set was based on—indeed was essentially identical to—that of Lotus 1-2-3. Equally true, the Copyright Act defines a derivative work as one "based upon one or more preexisting works." 17 U.S.C. § 101 (definition of "derivative work"). However, even if the Lotus menu command hierarchy were copyright subject matter and given that Borland based its emulation mode command set on that of Lotus, such a taking cannot infringe an assumed derivative work right without a showing that Borland took
Consequently, we should be cautious—certainly more cautious than essentially any court has yet been—before jumping into an analysis of the degree of copyright protection for interfaces, under the Computer Associates or any other analytical method for separating protected from unprotected elements of works. Under the analysis provided herein, the interface is not a nonliteral element of the program, and the statutory definition of a computer program does not allow it to be so considered except by judicial fiat. Under the statutory definition, essentially all nonuser interfaces lie outside the realm of copyright protection, and aspects of user interfaces such as keyboard or other input-output operations will often fall into that category as well. Still, many user-interface disputes have involved, and will continue to involve, screen displays, which can be pictorial or graphic works in their own right.\footnote{165} The analysis in the next section shows that the limitations of Baker v. Selden and § 102(b) apply, however, to functional aspects of screen displays. Moreover, even if we assume, hypothetically, that nonuser interfaces or such aspects of user interfaces as the Lotus menu command hierarchy do somehow constitute copyright subject matter; that same analysis applies with equal force to deny copyright protection to their functional aspects.

B. Protection of User Interfaces as Independent Works

As discussed above,\footnote{166} the only thing that can be said with certainty about the congressional intent in bringing functional computer

\footnote{165. Infringement of screen displays was not an issue before the First Circuit in Lotus. 49 F.3d at 812.}
programs under copyright protection is that CONTU, and Congress, saw a need to protect program code from fast, cheap, and easy misappropriation. That § 102(b) and *Baker v. Selden* cannot apply in full force to code is clear if the goal of protecting code, especially directly functional object code, is to have any practical meaning. We must assume, therefore, that Congress intended that these fundamental limitations on the scope of traditional copyrights no longer apply to code. That assumption is supported by CONTU’s identification of code—and nothing else besides code—as in need of protection.¹⁶⁷ In the absence of any convincing policy basis for allowing copyright to trench further into the realm of patent subject matter, we must not lightly assume that Congress intended to go beyond this very significant break with tradition. Indeed, absent some evidence that Congress intended to go further, we must assume the contrary.

Consequently, if any aspect of an interface is protected by copyright, whether because it independently qualifies as a work of authorship or whether we simply accept the textually unjustifiable but nearly universally followed judicial approach of treating interfaces as nonliteral elements of the program itself, we must apply all of the traditional limitations on copyright protection to such works.¹⁶⁸ Copyright and patent have very different thresholds for, as well as scope and terms of, protection. Courts should not extend copyright by protecting functional¹⁶⁹ works without explicit congressional authorization. This is the basic teaching of *Baker v. Selden*.¹⁷⁰ That Selden’s accounting forms may have satisfied the minimal level of creativity required by the copyright law or could have been expressed in a variety of ways was irrelevant to the *Baker* decision.¹⁷¹ A copyright in the book in no way

¹⁶⁷. See supra notes 45-46 and accompanying text.

¹⁶⁸. Again I find myself apparently in at least partial agreement with Professor Patry, who has formulated the issue in *Lotus* as follows—if one adopts the view that the Lotus 1-2-3 menu command hierarchy is a compilation: “T]he issue [is] whether a compilation that admittedly contains expressive material is disqualified from protection because it performs a useful function.” Patry, supra note 54, at 16-17.

¹⁶⁹. See supra notes 4-32 and accompanying text.

¹⁷⁰. 101 U.S. 99, 102 (1880); see supra note 32.

¹⁷¹. Professor Patry apparently believes that § 102(b) does not apply to any aspect of a copyright-protected work that is creative. Patry, supra note 54, at 13. He argues that, instead of focusing on § 102(b) in determining the protection of nonliteral elements of programs, we should determine whether the program is independently created and possesses sufficient creativity so as to constitute an original work of authorship. In fact, he treats § 102(b) as an unnecessary appendage that only duplicates the originality requirement. See id. at 14, 36 (stating that “section 102(b) permits the use of unoriginal component parts” and that “the inclusion of section 102(b) was unnecessary. In the case of computer programs, the requirement of originality would have limited protection to the programmer’s expression even in the absence of section 102(b).”). In a similar vein, he argues that *Baker v. Selden* stands for no more than noncopyright protection of nonoriginality. See id. at 45-49. He does not consider the effect of patent law
prevents others from using the methods described, or the forms needed to execute the system.\textsuperscript{172} In this way, the Supreme Court established a critical limitation on copyright protection not only to ensure that technological advances not satisfying the exacting requirements of patent law do not indirectly receive protection through copyright, but also to preserve the balance and integrity of the entire intellectual property system.

Although the first court to decide an interface case got things right,\textsuperscript{173} later district courts in the 1980s addressing the problem of computer user interfaces—resting upon an erroneous assumption that the copyright in the computer program extends to the outputs of the program or building upon the faulty foundation of *Whelan*—went astray of these fundamental copyright doctrines.\textsuperscript{174} The winds began to change, however, with the district court’s opinion in *Apple Computer, Inc. v. Microsoft Corp.*\textsuperscript{175} The Apple court quite explicitly accepted the notion implicit in § 102(b) and in *Baker v. Selden* that functionality trumps copyright protection—that when something is functional, copyright protection can attach only to separable artistic features.\textsuperscript{176} It also recognized that the fundamental purpose of a user interface is to permit people to use the program quickly and flexibly for its intended purpose.\textsuperscript{177} Although the Apple court did not rely on the definition of a computer program to limit or deny interface protection, it did reject *Whelan* in favor of *Computer Associates*, succinctly describing the appropriate policy as one of balancing antipiracy rules against the value of allowing development by later creators: “[T]he *Whelan* rule distends copyright protection, placing off-limits alternative and improved means of expression and thereby upsetting the uneasy balance which copyright attempts to maintain by preventing free riders from ripping off creative

\begin{footnotes}
\item[172] Many cases have implemented this aspect of *Baker v. Selden*. For a more extended discussion and citations, see Samuelson, * supra* note 29, at 226-27 & n.73.
\item[173] Synercom Tech., Inc. v. University Computing Co., 462 F. Supp. 1003, 1014 (N.D. Tex. 1978) (holding data input formats unprotected, allowing defendant to create an independent program that would accept data written for use with plaintiff’s program).
\item[175] 799 F. Supp. 1006 (N.D. Cal. 1992), aff’d in part, rev’d in part, 35 F.3d 1435, 1439 (9th Cir. 1994), cert. denied, 115 S. Ct. 1176 (1995). The Ninth Circuit’s affirmation in Apple relied heavily on a license agreement between the two companies, which enabled it to avoid discussing some of the more basic problems of interface protection. This does not detract from the persuasive force of the district court’s reasoning, however.
\item[176] See 799 F. Supp. at 1021, 1023, 1027 n.19, 1039.
\item[177] See id. at 1023.
\end{footnotes}
expression while not stifling others from improving or extending that expression." 178

The reasoning of the district court in Apple stands in stark contrast to that employed by Judge Keeton at the district court level in Lotus. 179 The series of opinions involving the Lotus 1-2-3 spreadsheet provide a good example of how judicial logic can lose its real-world bearings. 180 To be sure, much of Judge Keeton’s analysis is more sophistry than logic. 181 As a basic policy matter, perhaps, his fundamental premise that functionality in the digital context does not preclude copyright protection is a value judgment that cannot be proven right or wrong, but it is inconsistent with over one-hundred years of copyright tradition beginning with Baker v. Selden, at least if one understands the term “functionality” in the sense that distinguishes patent and copyright subject matter. 182 Nor is this tradition overcome by the congressional decision to treat computer programs as copyright subject matter. The Lotus decision provides no grounds for its assertion that Congress intended to throw away the Baker tradition in adopting the CONTU recommendation. 183 Judge Keeton’s failure to understand this tradition,

178. Id. at 1025. Although the Ninth Circuit relied heavily on the license agreement in upholding the lower court’s finding of noninfringement, the decision shows a good understanding of the need to be cautious against overprotecting functional works. See 35 F.3d at 1442. It adopts, for example the “virtual identity” standard for infringement by illicit copying “[w]hen the range of protectable and unauthorized expression is narrow.” Id. at 1439. The court conceded that a user interface can be partly artistic and partly functional, but even the artistic creativity is constrained by hardware characteristics, and “[d]esign alternatives are further limited by the [graphical user interface’s] purpose of making interaction between the user and the computer more ‘user-friendly.’” Id. at 1445.


181. Judge Keeton’s argument that his three-part test for copyrightability is compatible substantively, though not entirely in methodology, with Computer Associates, is a ground for the critical comment. See 799 F. Supp. at 215-16. His second step, for example, considers only the number of ways an idea, system, or process can be expressed. See id. at 217. This second step is the only one in which Judge Keeton filters out unprotected elements, so it bears no resemblance to filtration in Computer Associates, which considers, in addition to merger and scene a faire, efficiency, compatibility, § 102(b), and the traditionally excluded items under Baker v. Selden.

182. See supra notes 2-38 and accompanying text.

183. In its earlier 1992 opinion, the court explained at greater length why functionality or a finding of “process” would not preclude copyright protection. The court there correctly stated that all computer programs are functional and that every aspect of a program is part of a process, yet Congress has mandated that something in computer programs be protected. See 788 F. Supp. at 90-93. It therefore rejected Borland’s argument—that “process” defeats “copyrightability” regardless of the originality of any particular expression—as “fundamentally inconsistent with the congressional balance struck in the Copyright Act” and went on to downplay the importance of Baker v. Selden as well as the views of the CONTU Report. Id. at 90-93.

Unfortunately, the court need not, and should not, have gone so far. That Congress has mandated some program protection says nothing about where the balance is to be drawn. Nor is there any
and the potentially radical deviation from it we would be making were we to protect computer programs and their related functional interfaces so broadly, led him to start his logical analysis from an incorrect premise.

On a technical analytical level, a court could hardly face a situation in the modern age more squarely on point with Baker v. Selden than Lotus v. Borland. Selden had created an accounting system that employed blank forms, the organization and content of which were necessary for anyone to use Selden’s system. The forms were, in other words, the predigital interface between Selden’s system and the user. Just as Selden’s accounting system was an incremental improvement upon prior systems of accounting, Lotus freely built upon prior electronic spreadsheet systems dating back at least to VisiCalc, introduced in 1979.\(^{184}\) Like Selden, Lotus started with existing technology and created a system for entering information into a computer, processing the information, displaying the information on the screen, and putting results obtained into hard copy.\(^{185}\) That system is implemented by a computer program and, because of the speed of digital processing it is more sophisticated than Selden’s accounting system, but there is no essential distinction between them as far as traditional copyright law is concerned.

Judge Keeton rejected the argument that the precise menu commands and menu structure were not copyright protected because they are necessary to functional compatibility for the following reasons:

---

\(^{184}\) See Menell, supra note 128, at 1057 (discussing the process of technological advancement in application programming). The basic methods underlying VisiCalc and Lotus 1-2-3 extend back at least to the time of Luca Pacioli, as a principal application of spreadsheet analysis is bookkeeping. The key distinctive feature of the VisiCalc and Lotus products is the ability simultaneously to calculate and alter tables of information. The basic means by which this ability was brought into accounting—digital processing—was of course the product of neither VisiCalc nor Lotus but rather hundreds of predecessors, human and corporate, active in the development of computer science.

\(^{185}\) What the user creates using the Lotus system may, of course, be copyright protected. The user-created products of most spreadsheet and word processing software almost always have no function other than to inform or entertain. They are therefore not functional in the sense used to distinguish patent and copyright subject matter. This product of the end user must be distinguished from the functional interface developed by Lotus to assist the user’s creation of such works. The typewriter is a device that also assists in the creation of copyright-protected works, but the machine itself is not protected by copyright, nor is its user interface—the arrangement of the keys.
This argument is essentially tautological. As applied to any case involving a useful article, an argument of this kind would always define the idea to incorporate all the specifics of the particular expression of that idea in the allegedly copyrightable work. Nothing would be copyrightable under this methodology of analysis.\textsuperscript{186}

Paradoxically, Judge Keeton's analysis here is perfectly correct. Nothing in the user interface of Selden's accounting system was copyright protected, precisely because of the necessity of using that specific interface in practicing Selden's system. Unfortunately, Judge Keeton drew exactly the wrong conclusion from his analysis. His insistence that something in the Lotus 1-2-3 system must be copyright protected led to a result in direct conflict with the holding of \textit{Baker v. Selden}.

The second step of Judge Keeton's infringement standard compounds this error by treating the problem as if it were simply one of merger. He found that by changing the names of the commands (and therefore the "natural" keystrokes used to invoke them, such as their initial letters) in the main menu and submenus, an "extremely large" number of possibilities different from the specific choices made by Lotus were possible.\textsuperscript{187} Had it been dealing with a fanciful work, the court's analysis would have been appropriate, but \textit{Baker v. Selden} and § 102(b) of the Copyright Act mean more than the merger doctrine for functional features.\textsuperscript{188} The whole point of \textit{Baker}'s distinction between patent and copyright subject matter is that systems and methodologies are not copyright protected, even if creative and even if other systems or methodologies are available. In this case, users of Lotus 1-2-3 have learned the Lotus system and many want to use it, not some other system. Other users have developed macros that must combine Lotus commands precisely as Lotus defines them in order to run on the Lotus system.

Applying Judge Keeton's analysis to the facts of \textit{Baker v. Selden} would therefore require a different outcome in that case. There are countless ways of bookkeeping and methods for inputting and processing accounting information. Hence Selden would be entitled to protection for his system as well as his forms for recording the information in the manner demanded by his system. Had Selden implemented his accounting system by way of a computer program, as Lotus did, rather

\textsuperscript{186} 799 F. Supp. at 217.
\textsuperscript{187} Id. at 217-18. The \textit{Lotus} district court adopted a similar approach with respect to the arrangement or hierarchy of the commands. \textit{Id.} at 218.
\textsuperscript{188} See supra notes 84-85 and accompanying text.
than simply describe it in a book, he would, under this analysis, be entitled to protect the "ruled lines and headings of accounts" and structured methodology of his system. Thus, the district court in this case effectively overruled Baker v. Selden, a central pillar in our intellectual property protection system.

This is not to say that there can never be copyright-protected expression in a computer program user interface. A straightforward example would be the fanciful characters and the environments in which they chase or avoid enemies in video games. It is even possible that some aspects of the Lotus 1-2-3 user interface, at least its screen displays, are included solely for aesthetic purposes and are not part of any of the processes, operations, or methods by which the user actually employs the program. Textual information under the "Help" operation will in many cases be copyright protected like any other literary work. But that should be the test: Is an aspect of the interface chosen solely to portray an appearance or to convey information or does it have another utilitarian function—is it something that the user must do in order to make this particular program operate in the intended manner? If the latter, § 102(b) and Baker v. Selden require that it be excluded from copyright protection.

Beginning with the First Circuit's reversal of Judge Keeton's decision in Lotus, the following subsections analyze subsequent user interface cases in the light of the analytical theory heretofore presented.

1. The First Circuit's Decision in Lotus v. Borland

As is now well known, the First Circuit in Lotus denied copyright protection in the Lotus 1-2-3 menu command hierarchy on the ground that it constituted a "method of operation" under § 102(b). The court correctly concluded that unprotected methods of operation were not, for example, limited simply to abstract, general means of interacting with a computer program that cause the program to perform spreadsheet calculations. Such an approach would incorrectly treat § 102(b) as if it means no more than merger by leaving any particular such means

---

190. See supra notes 2-38 and accompanying text.
192. 49 F.3d at 815.
copyright protected, at least if sufficient creativity went into its design and if a variety of other such means could be imagined. The First Circuit, rather, took § 102(b) to mean what it says: “methods of operation” are the specific methods of causing something to operate, so even the specific approach designed by Lotus falls outside of copyright. This allows the particular method of operation partially designed, but not patented, by Lotus, to be the subject of incremental improvement just like any other unpatented technological product.

The First Circuit’s opinion in Lotus is not as tightly reasoned as it could have been and may, therefore, leave some fine-tuning to later decisions. For example, although the Lotus court recognized that the program copyright was not relevant to protection of the user interface, it never sought to determine just what kind of copyright-protected work, if any, the Lotus menu command structure was. The command structure was, indeed, copied literally, but that is of no copyright relevance if the command structure is not a protected work. And because it is not a nonliteral element of the program, its only conceivable link to copyright protection would be as a nonliteral element of the collection of interface screens (which are graphic works, and perhaps even audiovisual works or compilations). Of course, the Lotus court did not have to decide this subject-matter question, because as a method of operation under § 102(b), the command structure could not, in any event, be protected.

Later courts dealing with asserted copyrights in screen displays, however, will often have copyright subject matter before them. These courts must avoid the temptation to leap from the subject-matter determination to the erroneous conclusion that every creative aspect of the screen displays is protected. They must inquire into the degree to which screen design—the placement of icons and instructions, for example—is aimed at ease or efficiency of use, in which case copyright protection should be denied as a method of operation under § 102(b), or purely as an appeal to the esthetic appreciation of the user, in which case copyright protection is appropriate. In other words, screen displays, like all other aspects of interfaces, are not covered by the program copyright; therefore, all of the traditional limitations on

193. Id.
194. Lotus itself did not start from ground zero in designing its interface but rather built on its predecessors, particularly VisiCalc. See Menell, supra note 128, at 1057.
195. “In the instant appeal, we are not confronted with alleged nonliteral copying of computer code. Rather, we are faced with Borland’s deliberate, literal copying of the Lotus menu command hierarchy.” 49 F.3d at 814.
196. See supra notes 154-65 and accompanying text.
197. See supra notes 53-66 and accompanying text.
copyright protection, including the doctrine of *Baker v. Selden*, continue to apply. This is clearly implicit in the *Lotus* decision, but it could have been stated more clearly.

Another point that remains implicit in *Lotus* is the relationship between patent and copyright and, in particular, how far Congress intended to allow copyright to trench upon the traditional domain of patent by including computer programs under copyright. For example, *Lotus* holds menu command hierarchies copyright unprotected as "methods of operation," but it never expressly recognizes that computer programs themselves are also "methods of operation," which are undeniably protected by copyright. The answer, clearly, is that by placing program code under copyright, Congress intended to that extent to override both *Baker v. Selden* and § 102(b). 198 How much further, if at all, Congress intended copyright to encroach into the traditionally forbidden area of functionality 199 is the problem addressed by *Computer Associates*. The *Lotus* court did not have to address this question either, because it realized that it was dealing not with a program, but rather with the product of a program. This meant that all of the traditional limitations on copyright protection continued to apply. Still, the key to resolving most of the interpretive problems of applying copyright to computer programs lies in recognizing the complementary but very different roles played by patent and copyright in the overall intellectual property protection scheme. Ultimately, the courts will have to recognize these different roles more explicitly.

Despite minor shortcomings, the First Circuit's decision in *Lotus v. Borland* brings us much closer to what must in the long run be the ultimate solution to the interface-protection problem: absent congressional action, there can be no copyright protection for functional aspects of program interfaces or their design, no matter how creative.

---

198. That is also the reason for treating programs as literary works and not subjecting them to the separability test applicable to pictorial, graphic, and sculptural works. The similarity of programs to traditional literary works is purely formal, but the separability test would deny copyright protection even to code, contrary to clear congressional intent. See *supra* text following note 52.

199. Recall that functionality, in the sense that distinguishes patent from copyright, means something more restricted than simply useful. See *supra* notes 5-21 and accompanying text. Judge Boudin's concurring opinion in *Lotus* makes the mistake of conflating the informational "utility" of a dictionary (long protected, although by a thin copyright) with the technological "utility" of a computer program. 49 F.3d at 819-20. The utility of a computer program is the kind of functionality that has always been reserved for patent law and would not be copyright protected at all under § 102(b) but for the explicit congressional exception for computer program code.
2. Other Circuit Court Decisions

At roughly the same time the First Circuit was considering *Lotus v. Borland*, the Fifth Circuit decided *Engineering Dynamics, Inc. v. Structural Software, Inc.*. This case involved data input formats for structural analysis programs. The court concluded without analysis that the user interface was a nonliteral element of the protected program; and, it had little trouble determining that the formats taken as a whole were expression because of their role in mediating between the program and the user, identifying the information required, and how it was to be ordered. The court showed no recognition that the formats not only inform the human user but also, once the user has assembled and ordered all the data, the program. Consequently, no competing program can make use of the user's already assembled and entered data without using essentially the same formats. The issue here is identical to the inability of Borland's purchasers to make use of their Lotus macros with a competing spreadsheet unless the menu command hierarchies are identical. *Engineering Dynamics* relies heavily on the district court opinion in *Lotus v. Borland* and makes the same analytical mistake: it concludes that creativity in methodological or systemic features of a program interface may of itself lead to copyright protection.

The other courts of appeal that have considered the user interface problem have shown a much greater sensitivity to the need to limit copyright protection of functionality. The Ninth Circuit in *Apple*, for example, correctly recognized that a user interface could have creative and copyright-protected artistic aspects, but it was express in treating user friendliness as a design constraint. Certainly macro and data format compatibility between competing programs is an important aspect of user friendliness, not to mention the ability of users to switch programs to which they have become accustomed without having to

200. 26 F.3d 1335 (5th Cir. 1994), supplemented, 46 F.3d 408 (5th Cir. 1995).
201. See supra note 173. In fact it seems that the same company that won as a defendant in *Synercom*, which held that the input formats were not copyright protected, won again as a plaintiff here when the Fifth Circuit held that its input formats were protected against taking for compatibility by yet a third competitor in the structural analysis program market!
202. See 26 F.3d at 1341.
203. See id. at 1344.
204. The Fifth Circuit in this case unwittingly recognized the importance of incremental improvement in the context of user interfaces by finding creativity in plaintiff's interface partially on the basis that defendant's program managed to perform the same functions with significantly fewer input formats. See id. at 1346. It should have realized that creativity in plaintiff's unpatented but functional product should not be a basis for long-term prohibition under copyright of defendant's further exercise of creativity in improving the product.
205. See supra note 178.
spend hours or days learning a completely different set of commands and features. Similarly, although the Eleventh Circuit in Bateman v. Mnemonics refused to hold that interface specifications are categorically outside copyright as a matter of law, the court also recognized the importance of compatibility concerns in the infringement analysis and expressly denied protection for the "functional results" of the execution of program instructions. 206

The Eleventh Circuit went even a bit further in upholding the district court's finding of noninfringement of a wood truss layout program in MiTek Holdings, Inc. v. Arco Engineering Co. 207 Although the court treated the user interface as a "nonliteral element" of the program, 208 it held the menu structure unprotected as a method of operation under Lotus 209 and explicitly recognized the different roles for patent and copyright in the protection of computer programs:

MiTek seems to misapprehend the fundamental principle of copyright law that copyright does not protect an idea, but only the expression of the idea. The idea-expression dichotomy is clearly set forth in . . . § 102(b) . . . . Were we to grant copyright protection to MiTek's user interface, which is nothing more than a process, we would be affording copyright protection to a process that is the province of patent law. As the Federal Circuit stated, "Patent and copyright laws protect distinct aspects of a computer program." Atari Games Corp. v. Nintendo of America, Inc., 975 F.2d 832, 839 (Fed. Cir. 1992). Patent law "provides protection for the process or method performed by a computer in accordance with a program," whereas copyright protects only "the expression of that process or method." Id. If, however, the patentable process and its expression are indistinguishable or "... inextricably intertwined," then "the process merges with the expression and precludes copyright protection." Id. at 839-40. Such is the case with the menu and the submenu command tree structure of the [MiTek] program. 210

The court thus reaffirmed the importance of the principles of Baker v. Selden in distinguishing between patent and copyright subject matter. 211

---

206. 79 F.3d 1532, 1547 & n.33 (11th Cir. 1996).
207. 89 F.3d 1548 (11th Cir. 1996).
208. Id. at 1555 n.15. This is contrary to the analysis suggested herein based on the statutory definition of a computer program. See supra notes 53-66 and accompanying text.
209. See id. at 1556-57.
210. Id. at 1556 n.19.
211. The MiTek court also adopted the Ninth Circuit's "virtual identicality" standard for infringement to the extent that aspects of the user interface could be considered a copyright-protected compilation. Id. at 1558. Moreover, the court suggested that trial courts might better avoid the abstraction process in separating idea from expression by requiring the copyright owner to specify the original and non-§ 102(b)-excluded elements it deems copyright protected. See id. at 1555. Courts in the Second Circuit have
Another potentially very important user interface case, Mitel, Inc. v. IQTel, Inc.,\(^{212}\) has been on appeal before the Tenth Circuit for over a year as this is written. The district court's opinion is discussed immediately below.

3. District Court Interface Cases

The user interface at issue in Mitel is on all fours with that involved in Lotus v. Borland. The programs in question operated "call controllers" for business telephones and provided the user with a set of input codes that were used to set various parameters of the system to meet the needs of the user. The plaintiff claimed copyright protection in its set of input codes. The parties did not dispute that a variety of combinations or

actually dismissed infringement cases where the plaintiff refused to state with specificity just what was claimed as protected and what was not. See, e.g., Fonar Corp. v. Deccaed Servs., Inc., 983 F.2d 427 (2d Cir. 1993) (vacating a preliminary injunction that was insufficiently specific for failure of the copyright owner to produce a copy of the allegedly infringed software); Fonar Corp. v. Magnetic Resonance Plus, Inc., 920 F. Supp. 508, 519 (S.D.N.Y. 1996) (granting defendant's motion for summary judgment for failure of the copyright owner to give a meaningful definition of the work in question or of its protected elements). In a later case, the Second Circuit cut back somewhat on the scope of Deccaed, by stating that insufficient specificity to support an injunction against ongoing activity by the defendant need not be fatal to the plaintiff when the defendant is moving for summary judgment. See Fonar Corp. v. Domenick, 105 F.3d 99, 103 (2d Cir. 1997). Even in this case, however, the court emphasized that the trial judge has tools to compel the requisite disclosure to prevent trial by ambush. See id. The specificity of precisely what is protected and what is not, of course, is a characteristic of patent law, which requires for infringement the adoption of each of the claims of a patent, or their substantial equivalents. One of the problems of using copyright, with its vague substantial similarity standard for infringement, to protect technology is that an incremental improvement of unpatented technology might still infringe. See supra notes 22-24 and accompanying text. These incipient judicial demands for more specificity in the program context may indicate an increased recognition by the courts that protecting technology under copyright cannot simply go forward in a "business as usual" manner applicable to traditional (nonfunctional) works.

212. 896 F. Supp. 1050 (D. Colo. 1995). The author served as a consultant to IQTel's counsel at the district court stage of the proceedings. A Tenth Circuit decision finally issued in Mitel as this Article was going to press, affirming the district court's denial of a preliminary injunction. Mitel, Inc. v. IQTel, Inc., 124 F.3d 1366 (10th Cir. 1997). The court explicitly refused, however, to follow the First Circuit's holding in Lotus that classification of a program element as a § 102(b)-excluded item requires denial of copyright protection. Id. at 1372. Rather, the court purported to hold to Alien's abstraction-filtration-comparison approach as refined by the Tenth Circuit in Gates Rubber, notwithstanding that Mitel involved protection of a user interface rather than a computer program. The court ultimately denied copyright protection for those portions of Mitel's command codes that identified the function to be carried out by the command, on the ground that they were chosen arbitrarily and, therefore, lacked originality. It denied protection for the remaining portions of Mitel's command codes, which assigned parameter values relevant to the command, on the ground that they were some a faire, determined by standard programming conventions or dictated by external functionality and compatibility requirements. Id. at 1372-76. Like Lotus, Mitel does not consider whether the set of command codes constitutes copyright subject matter. Also like Lotus, Mitel fails to consider the relevance of the Copyright Act's definition of a computer program in determining either the existence, or the scope of the copyright protection in a user interface. See supra notes 150-53 and accompanying text. Unlike Lotus, however, Mitel displays very little understanding of the role of Baker v. Selden and § 102(b) in using functionality to distinguish between patent and copyright subject matter.
sequences could have been used to implement the various choices available to the user.\textsuperscript{213} However, once chosen, the user was required to use the commands presented by the program. Without directly relying on the First Circuit's opinion in \textit{Lotus v. Borland}, the Mitel court concluded that the command codes were simply a "procedure, process, system, and method of operation by which the customer can match the call controller functions to the long-distance carriers' technical needs and the end-user's choices."\textsuperscript{214} Even on the assumption that the codes were a part of the computer program—an argument that even the plaintiff did not press—the district court also concluded that the command codes were an unprotectable "means to access or operate the program ..."\textsuperscript{215} A contrary holding would have helped Mitel further entrench its dominant market position in the market for call controllers because the sunk costs of users in learning the Mitel system would inhibit them from switching to a competitor, even to a competitor who offered a technologically improved product. Courts must be very careful in allowing copyright to inhibit competition in the market for copyright-unprotected products.\textsuperscript{216}

\textit{Productivity Software International, Inc. v. Healthcare Technologies, Inc.}\textsuperscript{217} involved programs designed to be used in conjunction with word processing software to improve the user's efficiency by automatically expanding abbreviations into the correct words or phrases. Applying the \textit{Computer Associates} analysis to nonliteral elements of the programs, the court concluded that plaintiff failed to identify any copyright-protected elements of the individual screen displays, because everything claimed was either unoriginal or dictated by considerations of efficiency. Moreover, the overall arrangements of the screen displays in the two programs were not substantially similar, especially in view of the "narrow copyright protection" afforded to works lacking a high degree of originality.

One other decision must be mentioned, even though it was decided as a compilation case rather than as one involving an program interface.

\begin{itemize}
  \item \textsuperscript{213} See id. at 1053. In fact, just as Borland in \textit{Lotus} had developed its own "native mode" interface for users who could pull away from or were not otherwise dependent for things like macro compatibility on the Lotus interface, IQTel had developed its own set of codes, with arguably improved functionality. See id. at 1052. IQTel's device, however, incorporated a Mitel translator that recognized the Mitel codes, so that installers who were familiar with the Mitel system or who had already written installation programs for the Mitel system could switch to the IQTel device without substantial investment in retraining or reprogramming. See id. at 1052-53.
  \item \textsuperscript{214} Id at 1055.
  \item \textsuperscript{215} Id.
  \item \textsuperscript{216} See supra notes 26-32 and accompanying text.
  \item \textsuperscript{217} 37 U.S.P.Q.2d 1036 (S.D.N.Y. 1995).
\end{itemize}
Compaq Computer Corp. v. Procom Technology, Inc.\textsuperscript{218} involved competition in the market for hard disk drives. Plaintiff manufactured hard drives for use in one of its computer lines in which computers could be equipped with software capable of giving warnings of potential problems. One such potential problem was hard-drive failure, and plaintiff made the decision to have the software actually monitor five, given harddrive parameters and to give a warning of possible failure if any of the five threshold values, previously read by the software from the hard drive, were exceeded. Defendant sought to compete in the hard-drive market and sold compatible drives for the same line of computers. Defendant included the same threshold values for failure warnings as plaintiff had used. Plaintiff argued that copying the five threshold values violated a copyright in the numbers as a compilation, and the court agreed.\textsuperscript{219} Because the threshold values were not chosen whimsically by plaintiff, but rather after careful investigation and analysis, it is difficult to see a basis for copyright protection after Feist.\textsuperscript{220} In addition, the threshold values constituted an important part of the interface between the software and the hardware device; therefore, copyright should also have been denied on functionality grounds. Essentially, under the court’s decision, competitors are prohibited from making use of unpatented information determinable from a widely distributed product without breaching any confidential relationships and useful in optimizing the performance characteristics of a technological device. That is not, and should not be, the role of copyright in our system of intellectual property protection.

\textbf{VI. REVERSE ENGINEERING OF PROGRAMS\textsuperscript{221}}

Intellectual property law represents a balance between policies favoring protection, such as basic fairness to creators of socially desirable works and incentives to their creation, and policies favoring nonprotection, such as the free flow of knowledge and ideas and the social advantages resulting from one creator’s building on the work of another. Because this balance must be struck differently for functional

\textsuperscript{218} 908 F. Supp. 1409 (S.D. Tex. 1995).
\textsuperscript{219} See id. at 1419.
\textsuperscript{221} This section is based on Reverse Engineering and Professor Miller, supra note 1, at 991-95.
works in comparison with nonfunctional works, patent and copyright
effect their respective protective schemes in very different ways.
Copyright does not give the copyright owner complete control over all
possible uses of the protected work.222

Yet, when a program is available only in magnetically or
electronically encoded object code, these theoretically unprotected
elements of programs are not readily extractable, because no human
being can examine and understand a complex program in this form.223
Knowledge of these unprotected elements is often a vital aspect of
creating interoperable programs and systems. Moreover, a fundamental
part of the copyright balance—especially in view of the long term of
protection—is that ideas and other unprotected elements in publicly
available copyright-protected works should be free for all to use and
build upon. All other forms of literary works exist in human intelligible
form, and anyone is free to take the unprotected elements from a
publicly distributed work without the permission of the author for the
purpose of creating new works, even competing works. For this reason,
reproduction of the work for the purpose of extracting unprotected
elements was never an issue for traditional (nonfunctional) literary
works.

Programs, however, cannot be "read" like books. In order for a
human being to understand the ideas and other unprotected elements
contained in the object-code form of a computer program, a
technological process known as "reverse analysis" is necessary. Part of
this process involves making a technical copy or translation of the
program through what is called "decompilation" or "disassembly." The
question for copyright law is whether such technical copies infringe,
even where no use is made of the copies other than to examine them for
the purpose of extracting their copyright-unprotected elements.

It is worth noting that, in contrast to traditional works, program
creators can have no legitimate complaint against reproduction of their
source code if they could be assured that the source code would not be

222. See Ralph S. Brown, Eligibility for Copyright Protection: A Search for Principled Standards, 70 MINN. L.
REV. 579, 588-89 (1985) "The right to control the use of a work, although granted to inventors, has never
been part of copyright except as performance may be considered 'use.' Indeed, the absence of a 'use right'
helps to justify the relatively casual approach to granting copyright...." Id.

223. Even if the electronic representation is written out in 0s and 1s—which in any event would of
course constitute a copy of the program—it is extremely difficult if not impossible even for a skilled
programmer to make sense of it. See Sega Amicus Brief, supra note 26. For an excellent description of the
entire reverse engineering process for programs, and the difficulties involved in effecting reverse
engineering, see Andy Johnson-Laird, Technical Demonstration of "Decompiation," 16 COMPUTER L. REP., 469
(1992); see also Andrew Johnson-Laird, Reverse Engineering of Software: Separating Legal Mythology from Actual
reinput into a computer and used. The only value in putting object code into human-readable form—given that reinputting for its intended use into another computer is prohibited—is to permit study of the program either for making other programs, or for making more effective use of the program in question. Computer programs are the technology for using computers, and limitations on source code distribution can seriously impede the flow of technological knowledge concerning their development and use. Program creators naturally want to use copyright to prevent study of their source code because this effectively maintains their monopoly not only over the copyright-protected elements of their programs, but also over unprotected ideas and methods. Understanding their desire for more protection than copyright has ever given to any other form of technology, however, is no reason to give in to it.

The second factor of the fair use analysis in § 107 is critical for the reverse engineering problem. Computer programs differ from traditional copyright subject matter in that they are inherently functional works that directly cause machine processes to be performed (unlike maps or recipe books, which merely convey useful information to humans). As argued above, computer programs have properly been brought under the copyright umbrella, notwithstanding their inherent functionality, because their special vulnerability to piracy in the form of electronic copying of code distinguishes them from other works of technology. If the scope of copyright protection is properly limited, as also argued above, to insure that technological ideas remain free for other programmers to use, we should not allow their indirect copyright protection by blindly applying copyright concepts and language appropriate to traditional works but inappropriate to works of technology.

224. See New Protectionism, supra note 1; Dennis S. Karjala, Protection of Computer Programs under Japanese Copyright Law, 8 EUR. INTELL. PROP. REV. 105, 109 (1986); see also M. Lehmann & T. Dreier, The Legal Protection of Computer Programs: Certain Aspects of the Proposal for an (EC) Council Directive, COMPUTER L. & PRAC., Jan.-Feb. 1990, at 92, 95. These authors use similar reasoning to conclude, in Germany prior to the EC Software Directive and in the absence of a general fair use provision, that courts should treat interim copies made in the course of reverse analysis as mere technical copies and not as infringing reproductions. Id. They argue that the copyright goal is to assure the author adequate participation in the economic exploitation of her work. Id. Because decompilation for the purpose of extracting unprotected elements is not the form of exploitation envisioned by the prohibition against copying, a technical copy made in the course of reverse analysis should not be deemed to infringe the reproduction right. Id. In other words, not every “copy” should be deemed a “reproduction.” The analysis can be less strained under a general fair use provision, such as § 107 of the U.S. Copyright Act.


226. See supra notes 39-47 and accompanying text.

227. See supra notes 67-71 and accompanying text.
All other copyright-protected works carry their ideas and other unprotected elements on their face. Anyone is free to use these unprotected elements without the permission of the author, even in works that directly compete. No fair use question arises because it is not infringement simply to extract unprotected elements. Computer programs in object-code form are alone, among all types of publicly distributed copyright-protected works, in being unreadable by human beings. Unless the program can be put into an intelligible form, it cannot be studied as a whole. Without studying the program as a whole, its unprotected ideas and processes cannot be discerned, nor can unprotected elements embedded in programs, such as interface information, be detected as such. One of the principal justifications for the very long period of copyright protection, the absence of general compulsory licensing provisions, and the automatic subsistence of protection without any independent review of novelty, is that the elements needed by later authors to advance cultural development by building on the past are precisely those that are unprotected by the copyright. If access to unprotected elements in publicly distributed works can be technologically denied and enforced through copyright, the fundamental copyright equation is thrown out of balance.

This basic argument in favor of permitting reverse analysis of programs for the purpose of extracting and using unprotected elements has been made in several forums and has now been strongly confirmed by Atari Games Corp. v. Nintendo of America, Inc. and especially Sega Enterprises Ltd. v. Accolade, Inc. At least one federal district court has agreed with this analysis. Many other courts have cited these

228. Of course, legal access to the work is usually necessary in order to extract ideas or other unprotected elements without violating some other legal rule, but legal access is always available when a work has been publicly distributed. But see ProCD, Inc. v. Zeidenberg, 86 F.3d 1447 (7th Cir. 1996) (upholding a shrinkwrap license prohibiting the taking of factual information from a publicly distributed compilation of telephone numbers). I argue elsewhere that ProCD was wrongly decided insofar as it upholds state contract law terms expanding the copyright owner's federal copyright rights in widely disseminated works. See Dennis S. Karjala, Federal Premption of Shrinkwrap and On-Line Licenses, 22 U. DAYTON L. REV. 511 (1997).

229. See Sega Amicus Brief, supra note 26; LaST Frontier Conference, supra note 128.

230. 975 F.2d 832 (Fed. Cir. 1992). The author served as a consultant to counsel and potential expert witness for Atari Games in this litigation.

231. 977 F.2d 1510 (9th Cir. 1993).

232. See DSC Communications Corp. v. DGI Techs., Inc., 898 F. Supp. 1183 (N.D. Tex. 1995), aff'd 81 F.3d 597 (5th Cir. 1996) (legitimate interest in gaining access to unprotected aspects of firmware in order to compete with a compatible product justified making copies of plaintiff's copyright-protected operating system to the extent necessary for testing).
cases with apparent approval. Consequently, it appears that reverse analysis when necessary in extracting copyright-unprotected elements from a protected work is, and is likely to remain, a noninfringing fair use.

VII. CONCLUSION

The protection of computer software under copyright law need not be so complex as the courts have made it. A coherent theory of software protection begins simply from a recognition of functionality, properly defined, as the crucial distinction between copyright and patent subject matter. Sensible conclusions then follow naturally from two axioms: First, the statutory definition of a computer program as a set of statements or instructions to be used in a computer to bring about a certain result implies that what the program does—the certain result its execution brings about—is not covered by the program copyright. Second, the policy basis for deviating from the patent model for this particular class of functional works is the vulnerability of program code to fast, cheap, and easy misappropriation.

The conclusions that follow are: First, the program copyright has a narrow scope of protection limited to literal code and fast, cheap, and easy derivatives from code. The protection of code functionality is not limited by § 102(b) or the doctrine of Baker v. Selden, although traditional limiting doctrines like merger, scenes a faire, and nonoriginality apply in the usual way. More abstract nonliteral elements of programs, such as structure, sequence, and organization of the program or its data files are not protected by copyright at all. Those elements must find intellectual property protection under patent or trade secret law. Second, program interfaces, including user interfaces, are copyright protected, if at all, only by an independent copyright as a traditional class of copyright-protected work, such as a pictorial, graphic, or ordinary literary work. All of the traditional limitations on the scope of protection, including, in particular, § 102(b) and the doctrine of Baker v. Selden, apply with full force. This approach eliminates most of the metaphysics from the otherwise intractable job (for complete want of standards) of separating idea from expression in creative but functional works.

These conclusions make sense in terms of policy, in terms of the statutory language, and in terms of our long tradition of intellectual

property protection as a coherent mix of patent, copyright, trademark, trade secret, and unfair competition law. The courts, under the leadership of the Second Circuit's decision in *Computer Associates* and the First Circuit's decision in *Lotus v. Borland*, are gradually honing in on these conclusions, as seen in the results of individual cases. However, no court has yet articulated the theory for protecting computer programs under copyright that leads to these results in a smooth and natural way. The approach given herein provides that theoretical path.