COPYRIGHT IN COMPUTER PROGRAMMING LANGUAGES

RICHARD H. STERN*

TABLE OF CONTENTS

I. Introduction .................................... 322
II. The *Paperback* Decision ........................ 324
   A. Programming Languages Defined ............ 327
   B. What the *Paperback* Court Protected .... 328
   C. Nonliteral Copying and Idea/Expression ..... 331
III. Case Law on Languages .......................... 333
    A. *Reiss v. National Quotation Bureau* .... 334
    B. *Hartfield v. Peterson* ................... 338
    C. The Shorthand System Cases ............... 340
    D. *Ashton-Tate Corp. v. Ross* .............. 342
    E. *NEC Corp. v. Intel Corp.* ............... 344
    F. Comparison of Decisions in Field ......... 346
IV. Refining the Question ............................ 347
    A. Languages Taught by Books and Compilers .. 348
    B. Languages of Other Computer Programs .... 355
V. An Approach to Programming Languages Based on *Paperback* .............................. 363
    A. Idea and Expression in Languages .......... 363
    B. Purpose as Idea ................................ 364
    C. Functional Aspects of Programming Languages ............................................. 369
VI. Using Copyright to Encourage Technological Advance ...................................... 371
    A. Institutional Considerations ............. 372
    B. Interests of Users ........................... 373
    C. Compromise Solutions ........................ 375


* Richard H. Stern Member, District of Columbia Bar, Connecticut Bar,
U.S. Patent & Trademark Office Bar. Formerly Chief of the Intellectual Property Section, U.S. Dep't of Justice and Visiting Distinguished Professor of Law, U. of Minn. Law School. I wish to acknowledge my appreciation of Professor Leo J. Raskind's helpful comments on earlier drafts of this article.
Until quite recently few observers would have considered copyright protection for computer programming language to be a matter of legal controversy, or even concern. The general assumption was that computer programming languages were not subject to copyright protection because they were unprotectable "ideas," rather than protectable "expressions" of ideas. Outside the United States, the general belief that such languages are unprotectable ideas still remains unquestioned.

1. For a discussion of the definition of the term "computer programming language," see infra note 13 and accompanying text. The computer programming languages with which this article is concerned are artificial languages used for writing computer programs or for other computer-related use. They are to be distinguished from natural, informal languages used for communication and discourse among persons. Existing natural languages, such as English or French, long predate the copyright laws. They have no specific author, and thus they are not original works of authorship that could be candidates for copyright protection. See 17 U.S.C. § 102(a) (1988) ("Copyright protection subsists, in accordance with this title, in original works of authorship."); see also Feist Publications, Inc. v. Rural Tel. Serv. Co., 111 S. Ct. 1282 (1991) (general discussion of constitutional requirement of originality of authorship for copyright to attach to work).

2. "In no case does copyright protection for an original work of authorship extend to any idea, procedure, process, system, method of operation, concept, principle, or discovery, regardless of the form in which it is described, explained, illustrated, or embodied in such work." 17 U.S.C. § 102(b) (1988). The statute codifies the holding in Baker v. Selden, 101 U.S. 99 (1879). See infra note 26 and accompanying text. The term idea is often used to refer collectively to procedures, processes, systems, and the other things to which § 102(b) states that the protection of a copyright does not extend.

3. National copyright laws generally have not addressed the issue expressly. Probably the only exception is the Japanese copyright law. Copyright Act of 1970, art. 10, para. 3, amended by Law No. 62 (Japan 1985), provides that copyright protection "shall not extend to any programming language, rule, or algorithm." As defined subsequently in that paragraph, "programming language" means symbolic vocabulary and "their systems for use as a means of expressing a program," while "rule" includes syntax rules for programming languages. See generally Karjala, The Protection of Operating Software Under Japanese Copyright Law, 10 EUR. INTELL. PROP. REV. 359, 364-67 (1988).

The initial version of the European Community's Directive on the Legal Protection of Computer Software, which requires member states to protect computer programs by their national copyright laws, expressly excluded computer programming languages from protection, but the final version omitted reference
That assumption has recently been challenged by the trial court’s decision in *Lotus Development Corp. v. Paperback Software International*. As a result, the copyrightability of computer programming languages has become a highly controversial issue within the software industry. The shared general assumption that anyone could freely use any programming language has given way to uncertainty. If an interpretation of copyright law that unauthorized use of a programming language creates copyright infringement liability ultimately prevails, software vendors to the issue. *Amended Proposal for a Council Directive on the Legal Protection of Computer Programs*, 33 J.O. COMM. EUR. (No. C 320) 22 (1990) (comparing the original with the amended proposal). The initial version of article 1(3) excluded from computer program copyright protection “the ideas, principles, logic, algorithms [and] programming languages underlying” computer programs. *Id.* at 25. The final version of article 1(2) merely states that “[i]deas and principles which underlie any element of a computer program, including those which underlie its interfaces, are not protected by copyright under this Directive.” *Id.*


A false start to resolving the programming language issue occurred in Ashton-Tate Corp. v. Fox Software, Inc., 760 F. Supp. 831 (C.D. Cal. 1990). Ashton-Tate sued Fox for copyright infringement on the basis of unauthorized competitive use of an input-command language for database management programs. The court granted summary judgment against Ashton-Tate on the unrelated ground of alleged fraud on the Copyright Office in the copyright registration process, invalidating the dBase II-IV program copyrights. *Id.* The alleged fraud was failure to disclose to the Copyright Office that many of the command words used in dBase II were taken from JPLDIS, an earlier public-domain program. *See dBase Copyright Invalidated*, A.B.A. J., Mar. 1991, at 30, 31 (33 of 58 command words taken from JPLDIS). Four months later, the court reversed its prior order, without explanation, and reinstated Ashton-Tate’s copyright infringement action. The proceedings have been so bizarre that it is unlikely that any opinion in this case will illuminate the issue of copyright protection of programming languages.
and users will have to adapt their current practices to conform to a new mandate of the law.

The *Paperback* decision has thus turned a non-issue into a focus of major controversy. The decision has provided a precedent for the recognition of proprietary rights in computer programming languages. Rights may now be asserted in languages as copyrighted works, as such. Alternatively, proprietary rights may be asserted in computer programming languages as non-literal aspects of copyrighted computer program works that embody them. This article explores what is called, perhaps in oversimplified terms, the copyrightability of computer programming languages. The article begins with a discussion of *Paperback* and the prior case law relating to computer programming languages. The article then explores, in various factual contexts, the question of whether computer programming languages are protected under existing copyright law against use without consent. The article next considers whether programming languages can properly be considered "non-literal but expressive" aspects of computer programs using or embodying them. The article concludes by considering the policy consequences of various approaches to legal protection of computer programming languages.

II. THE *PAPERBACK* DECISION

In *Paperback*, the Massachusetts District Court expressly rejected the proposition that languages cannot be protected by copyright. The court stated that it was holding the defendants liable for copyright infringement because they had copied the "user interface" or "command structure" of the plaintiff's "1-2-3" spreadsheet program. The court did *not* consider those aspects of the program to be a computer programming language. The court added, however, that it could not agree with the defendants that a computer programming language, such as Pascal, was necessarily ineligible for copyright protection against infringers.

---

6. *Id.* at 68-70.
7. *Id.* at 72.
The *Paperback* court applied the terms *user interface* and *command structure* to refer to the set of input-command words that a user would employ to direct performance of tasks by the 1-2-3 computer program and to the rules or system for combining

8. The 1-2-3 spreadsheet program involved in the *Paperback* case is a computer program for performing specific tasks. Other examples of task-specific computer programs are the "WordPerfect" word processing program and the "dBase" database management program. Task-specific computer programs have two major aspects. One aspect is the "user interface," which is the means by which a user of the program interacts with it, such as for purposes of inputting data or commanding performance of tasks. The other major aspect is the "engine." See infra note 10.

9. Input of data and command of tasks can be effected by transmitting instructions to a computer program. One way to formulate instructions is in terms of input-command words combined together in accordance with rules. The set of input-command words and the rules for combining them constitute the *command structure* (or "input-command structure") for the computer program. Ordinarily, a user transmits instructions by pressing keystrokes on a keyboard, where given keystrokes represent given command words. A command structure is only one aspect of a user input interface. See infra note 11.

Use of keystrokes representing input-command words is only one way to effect input commands. An alternative method is to use a sequence of menu screens from which the user selects options by keystroke—usually moving a cursor to a desired choice and selecting that choice by entering a keystroke. This method permits the user to direct performance of tasks in a menu-driven mode instead of a command-driven mode. Usually the hierarchal relationship ("menu tree") of the program's set of menu screens and the hierarchal relationship ("command tree") of commands in the program's input-command language are isomorphic. In effect, they have the same syntax. Most task-specific programs permit users to employ either mode, since menu-driven operation is easier for novices and command-driven operation is faster and less tedious for experienced users. The 1-2-3 program involved in the *Paperback* case permitted both modes of operation, and the two trees were isomorphic.

Command structures are not unique to task-specific applications programs. Operating system software, such as MS-DOS and Unix, also permits input of commands by keying words into the computer or reading them into it from programs. For example, keying `DIR` into an IBM-compatible personal computer (when the `A>` or `C>`) prompt is on the screen) will cause a directory of files to appear on the screen. The command language for such MS-DOS computers is known as "DOS batch command language." Programs written in that language are known as batch programs. The most common and familiar form of them is an AUTOEXEC.BAT program, which users write to be able to make their systems start up in a desired configuration. Thus, an AUTOEXEC.BAT program with `DIR` as its last line would cause a file directory to appear on the screen after the computer was turned on.

10. Another major aspect of such a task-specific program is its "engine," which is a term for the underlying code for performing calculations or other-
the input-command words. This terminology obscured the legal issues and, doubtlessly unintentionally, disguised the significance of the court's extension of copyright protection to an input-command language.

The defendants did not copy the code of the 1-2-3 computer program; the codes of the two computer programs were quite dissimilar. The similarity of the two programs was in what they did and how they did it, not in their literal content. The 1-2-3 program enabled its users to direct performance of tasks by using a set of input-command terms. The defendants' program utilized the same input-command terms, in the same way. For example, users of each program would enter the keystrokes /FR (meaning "This is a Command for File Retrieval") to bring up a menu of files of spreadsheets. Although the engines of the two spreadsheet programs were different, the two engines responded to the same user commands in the same way. By way of a very simple analogy, a Plymouth and Ford have different engines, but each

wise carrying out the tasks for which the program is designed. See Ashton-Tate Corp. v. Ross, 916 F.2d 516 (9th Cir. 1990) (describing spreadsheet engine).

The engine of such a program will ordinarily be written in a first computer programming language, such as C or 8086 assembler, which is already in common use. The user interface of the program will embody or enable use of a second, higher-level language (input-command language). The latter is peculiar to that task-specific program, and is often developed for the particular application in question. As described in the text that follows, the Paperback court held that the copyright in the underlying program extended to the input-command structure of the user interface, meaning the words for making command inputs and the permissible patterns in which, or the rules or system pursuant to which, the words could be combined.

11. The user interface of a task-specific program has another function and component, not essential to the present analysis of programming languages and not addressed in detail here. User interaction with the program also includes its presentation (output) of information to the user, such as giving cues for user inputs and displaying results accomplished or tasks performed. Those user-interface functions are usually carried out by screen displays. For a discussion of technical aspects of screen displays and other user interfaces, see W. Galitz, HANDBOOK OF SCREEN FORMAT DESIGN (3d ed. 1988). For a discussion of legal protection of screen displays, see Stern, Legal Protection of Screen Displays and Other User Interfaces for Computers: A Problem in Balancing Incentives for Creation Against Need for Free Access to the Utilitarian, 14 COLUM.-VLA J.L. & ARTS 283 (1990). Similar issues of how to distinguish idea and expression pervade any copyright analysis of screen displays or programming languages.
car goes backwards if put in reverse gear; each speeds up when its accelerator pedal is depressed.

To find infringement of the copyright in the plaintiff's 1-2-3 computer program, therefore, the court could not rely on any literal similarity of the coding of the two computer programs. Instead, the court concluded that the command words and the rules or system for combining them (the command structure of the 1-2-3 user interface) constituted a "nonliteral" aspect of the computer program, and that the copyright in the program protected such nonliteral aspects of the program against unauthorized use. Despite the court's disclaimer, it effectively held the defendants liable for copyright infringement because they used a programming language without authorization.

A. Programming Languages Defined

What the Paperback court called a user interface and command structure clearly met the accepted definition of a programming language. The term "programming language" is not defined in the copyright statute or its legislative history. The term is defined, however, in electronics and computer dictionaries. Those definitions may be summarized as follows:

A computer programming language is a formal system of expression including:

1. a set of vocabulary elements;\(^{12}\)
2. a set of syntax rules for combining vocabulary elements into statements; and
3. the assignment of meaning to statements that properly combine vocabulary elements in accordance with the syntax rules.\(^{13}\)

\(^{12}\) Vocabulary includes operational elements such as AND, NOR, ADD, MOVE, PRINT, and COPY. Those elements operate on other vocabulary elements (data objects) such as x, y, 6, and 3.14159. Other vocabulary elements (control structures), such as GOTO . . . ; DO . . . WHILE . . . ; and IF . . . , THEN . . . , ELSE . . . , specify flow or sequence of operations.

As the following text will show, the protected elements of the 1-2-3 command structure include examples of each of the foregoing types of vocabulary element.

\(^{13}\) C. SIPPEL, MICROCOMPUTER DICTIONARY 212 (2d ed. 1981), gives this short definition of computer programming language: "A defined set of characters that are used to form symbols, words, etc., and the rules for combining these characters into meaningful communications." A similar but more de-
This definition probably corresponds, as well, to that of a formal system for symbolic logic notation.14

B. What the Paperback Court Protected

That the court in the Paperback case protected a computer programming language under the copyright laws is shown by analyzing what acts of the defendants the court condemned. The court said that it was protecting a "user interface" and identified that term primarily with "the choice of command terms" that a user of the 1-2-3 program utilizes in order to command the program—that is, to direct the engine of the program—to perform tasks.15 The court also said that it was protecting "the structure and order of those terms."16 At the same time, the court rejected as a mere "word game" the defendants' argument that they were not liable for copyright infringement because they had copied only an uncopyrightable "language."17 Analysis of the plaintiff's interest that the court protected, however, shows that it meets the accepted definition of a programming language.

The "choice of command terms" for 1-2-3 is the set of keystrokes and command words with which a user invokes functions of, or direct performance of tasks by the engine of the 1-2-3 program. The command words are terms such as "Print," "Move," or "Copy." A user may input them by keystrokes, usually the initial letters of the command words, such as P, M, or C. The set of all permissible command words is thus seen to be the vocabu-

tailed definition is found in IEEE STANDARD DICTIONARY OF ELECTRICAL AND ELECTRONIC TERMS 478 (3d ed. 1984):
A system consisting of: (1) a well defined, usually finite, set of characters: (2) rules for combining characters with one another to form words or other expressions: (3) a specific assignment of meaning to some of the words or expressions, usually for communicating information or data among a group of people, machines, etcetera.

The IEEE dictionary points out that there also exist "formal" or "uninterpreted" languages, which lack the third element—that is, an assigned meaning. Id. This article concerns only interpreted languages.

16. Id. at 68.
17. Id. at 72.
lary of 1-2-3, which is the first element in the definition of a programming language.

With few exceptions, a user does not employ a single command word or keystroke to direct performance of a task. Thus, the equipment cannot simply "Print"; it must be directed what to print, how or where to print it, and so on. Accordingly, to cause the computer system and the 1-2-3 program to cooperate to perform a desired task a user must input a series of command words in a proper or well-ordered sequence. Such a well-ordered sequence of command words is termed a "macroinstruction" or "macro." Unless the rules for ordering command terms are properly followed, the result is gibberish. It is meaningless to, and unrecognized by, the computer and the computer program. The set of rules for placing vocabulary elements of 1-2-3 in a proper or well-ordered sequence is what the Paperback court called the structure and order of command terms for 1-2-3. That is 1-2-3's set of syntax rules, the second element in the definition of a programming language.

Microinstructions that use the prescribed vocabulary and follow the syntax rules of 1-2-3 have an assigned meaning, the third element in the definition of a programming language. The Paperback court illustrated this with the following macroinstruction, which the court used to show how a user of 1-2-3 would input keystrokes to cause information in a portion of a spreadsheet to be printed:

/ P P R A 1 . F 1 9 {enter} A G Q

The symbols may be translated into ordinary English as: "This is a command to print the part of the spreadsheet ranging from coordinates A1 to F19, aligned."18

Each symbol set out above represents both a particular keys-

---

18. The statement as a whole means "Print the aligned contents of the cells in the rectangular area whose upper left corner is at coordinates A1 and whose lower right corner is at coordinates F19." The initial / means that a macroinstruction follows. The first P means that a printer command follows. The next P and the R mean that a print range follows. The A and 1 designate the coordinates of the upper left corner of the rectangle to be printed, and the period (.) is a delimiter indicating the end of this parameter entry. The F begins the designation of the next parameter entry, the coordinates of the lower right corner, and so on. Id. at 64-65.
troke and an input-command word or parameter that the keystroke signifies.\textsuperscript{19} Although the \textit{Paperback} court was unwilling to recognize it, the foregoing expression clearly constitutes a "sentence" or a "statement" in 1-2-3 language. The expression uses vocabulary (command terms) from the 1-2-3 vocabulary list; it follows the 1-2-3 syntax rules (structure and order of command terms); and it has a particular assigned meaning. The macroinstruction thus meets the criteria for a statement in a programming language. That is true, as well, for any other well-ordered macroinstructions of 1-2-3.\textsuperscript{20} By holding the defendants liable for copyright infringement because they appropriated 1-2-3's vocabulary elements and syntax rules, the \textit{Paperback} court, even if it did so unwittingly, protected the plaintiff against the defendants' unauthorized use of the plaintiff's input-command programming language.\textsuperscript{21}

\textsuperscript{19} Each command term is represented by a one-letter keystroke. But 1-2-3, like many programs, has more command terms than there are keys on a keyboard. Hence, the same keystroke or alphanumeric symbol (for example, F or R) may mean different things depending on its location within a macroinstruction. Thus, in the 1-2-3 macrocommand beginning /FR, F means "File" and R means "Retrieve." In the macrocommand beginning /RF, F means "Format" and R means "Range." (Compare the numbers 321 and 231. In 321, 3 means 300 and 2 means 20. In 231, 3 means 30 and 2 means 200.)

It is thus seen that 1-2-3 language uses a positional notation, in which symbols have different meanings if placed in different positions within the sequence of a statement. Similarly, "The dog ate the hat" and "The hat ate the dog" are both meaningful statements in English, but they have different meanings.

\textsuperscript{20} Not all sequences in 1-2-3 are well-ordered. Those that are not are meaningless. As indicated in the preceding note, both /FR and /RF begin meaningful commands in 1-2-3 language. But /FF and /RR do not. That sequence is not permitted by 1-2-3's syntax rules. Entering those keystrokes will cause a "beep" (tone) to be sounded, indicating user input error. Similarly, "The hat dog ate the" is not a meaningful statement in English, because it does not follow the syntax rules for word order in English.

The rules specifying permissible order for the entire set of permitted command terms in 1-2-3 may be represented with a tree-like chart, which is called the "command tree" of 1-2-3. The defendants in the \textit{Paperback} case used the same command tree for their spreadsheet as that of 1-2-3. Stern, \textit{The Paperback Case}, IEEE MICRO, Oct. 1990, at 9.

\textsuperscript{21} Lotus regarded its copyright infringement action against Paperback as being directed to the protection of a language. Jim Manzi, the CEO of Lotus, remarked that: "The lawsuits are targeted at two companies which . . . copied . . . menu structure and sequence, word selection, and macro language design."
C. Nonliteral Copying and Idea/Expression

Although the *Paperback* court did not acknowledge that it was applying copyright protections to the use of a programming language, it was quite aware that it was extending copyright protection beyond the literal copying of a computer program code. It held that the user interface or command structure of the 1-2-3 computer program was protected as "nonliteral" expression of the idea of the program.\(^{22}\) The court considered the idea of the 1-2-3 program to be to provide an "electronic spreadsheet."\(^{23}\) The court considered the command structure of 1-2-3 to be one of many possible ways to transmit input commands to an electronic spreadsheet, because different command structures could be devised for that purpose. In addition, the court considered the 1-2-3 command structure to be "original and nonobvious."\(^{24}\) For these reasons, it concluded that the 1-2-3 command structure was


\(^{23}\) The court used the term "electronic spreadsheet" to refer to a rectangular array of numbers displayed in boxes on the screen of a computer system and generated by a computer program. The numbers in some boxes have a predetermined arithmetic relationship to the numbers in other boxes. Because the system is computerized, the calculations necessary to cause the predetermined relationship to occur are performed by a computer program rather than by the user. The idea of such a product originated with Bricklin's earlier spreadsheet program, "VisiCalc." The authors of 1-2-3, Kapor and Sachs, decided to provide an improved version of VisiCalc and Kapor formed Lotus to do so. See Lotus Dev. Corp. v. Paperback Software Int'l, 740 F. Supp. at 65-66. Kapor designed most of the user interface; Sachs designed the remainder of the user interface and wrote all of the code for the engine. The court noted that most of the economic value of 1-2-3 is associated with its user interface. See id. at 68.

Given the last fact, it may be questioned whether the court accurately characterized the idea of 1-2-3 as simply the same old electronic-spreadsheet idea found in VisiCalc. At the very least, the idea of Kapor was to provide a more user-friendly spreadsheet. As will appear, a threshold determination of what the idea of a work is can determine the outcome of any copyright infringement analysis premised on that determination.

part of the protected expression of the underlying program, rather than an unprotected part of its idea.\textsuperscript{25}

Before addressing the problems that this legal analysis poses, more must be said about the distinction between idea and expression to which the \textit{Paperback} court referred. Central to any copyright infringement analysis of nonverbatim or nonliteral copying of a work (including, therefore, any analysis of whether languages are protectable under copyright law) is whether the subject matter that the accused work took from the copyrighted work is part of the unprotected "ideas," or part of the protectable "expression" of the ideas, of the copyrighted work. The copyright doctrine distinguishing idea and expression is derived primarily from the Supreme Court's landmark opinion in \textit{Baker v. Selden},\textsuperscript{26} and that decision's progeny.\textsuperscript{27}

\textsuperscript{25} \textit{Paperback}, 740 F. Supp. at 68.


In \textit{Baker}, the author Selden had published and copyrighted a book describing a new method of bookkeeping. Baker marketed some blank ruled forms similar to those published in Selden's book mainly in that both parties' forms facilitated use of Selden's new bookkeeping method. \textit{Baker}, 101 U.S. at 104. The Supreme Court held that Selden's copyright in the book protected only Selden's individual manner of describing his system—that is, the words with which Selden expressed his ideas about bookkeeping. The copyright did \textit{not} prevent competitive use of the method itself. Use of the art (method) of bookkeeping that the book taught, meaning use of Selden's ideas about how to keep books more effectively, was not prohibited by Selden's copyright. \textit{Id.} at 104-05. If the copyright were held to cover the use of the bookkeeping system, the copyright would have the effect of giving Selden a patent monopoly on his method of bookkeeping. But the safeguards of the patent system would not have been met as a precondition to allowing the monopoly. The Court therefore held Baker free to sell the bookkeeping forms without liability for copyright infringement. \textit{Id.} at 104-07.

\textsuperscript{27} In the present context, the most important of these cases is \textit{Morrissey v. Procter & Gamble Co.}, 379 F.2d 675 (1st Cir. 1967). This decision amplifies a point of copyright doctrine in \textit{Baker} concerning forms of documentation used in accomplishing given tasks. \textit{Id.} at 678. Whereas the \textit{Baker} case concerned blank forms for maintaining financial records in accordance with Selden's particular system of bookkeeping, the \textit{Morrissey} case involved the particular wording of rules for a contest. The contest rules in \textit{Morrissey} lent themselves to only few alternative wordings. When a particular form of documentation (expression) is one of only a few ways to carry out a particular idea (such as keeping
The *Paperback* court stressed that no prior decision had been cited to it explicitly holding that languages are ideas rather than expressions. Indeed, there is none. The court might equally have stressed that no decision exists that explicitly holds that languages are expressions rather than ideas; no court has as yet explicitly held either way. The time has now become ripe, however, for resolution of the issue. Several pending copyright infringement actions involve computer programming languages. Moreover, it is inevitable that other plaintiffs will assert claims to copyright in languages unless the *Paperback* holding is authoritatively rejected, and other defendants will assert that computer programming languages are uncopyrightable unless the theory is authoritatively upheld. It is therefore appropriate to examine the other copyright precedents relating to languages. Doing so will place the *Paperback* decision in context and help decide whether *Paperback* is consistent with copyright doctrine or is an unsupported aberration. That, in turn, will provide a foundation for exploring the question whether copyright law protects computer programming languages against unconsented-to use, in various factual settings.

### III. CASE LAW ON LANGUAGES

In the course of rejecting the defendants' claim that only an

books in accordance with Selden's system, or running a contest in accordance with Morrissey's scheme), copyright law does not protect the creator of the form of documentation against use by other persons. *Id.* at 678-79. Otherwise, as the Supreme Court indicated in *Baker*, the effect would be to give the copyright owner a monopoly over the idea. In such circumstances, it is said that the expression "merges into" or "merges with" the idea.

The *Morrissey* court held that idea-expression merger makes the subject matter uncopyrightable. *Id.* at 678-79. Other courts have held that idea-expression merger calls for a finding of noninfringement, on the ground that the scope of the copyright does not extend to any subject matter whose expression has merged with its idea or that similarity resulting from use of the same idea is not "substantial similarity" for copyright infringement purposes. *See, e.g.*, Data East USA, Inc. v. Epyx, Inc., 862 F.2d 204, 208 (9th Cir. 1988); Atari, Inc. v. North Am. Philips Consumer Elecs. Corp., 672 F.2d 607, 616 (7th Cir.), cert. denied, 459 U.S. 880 (1982); NEC Corp. v. Intel Corp., 10 U.S.P.Q.2d (BNA) 1177 (N.D. Cal. 1989).

unprotectable programming language had been copied, the *Paperback* court stated that it did not accept as a rule of copyright law that a language, such as Pascal, when original with a plaintiff, was "ineligible for copyright protection" against infringement. This is the closest that any court has come to explicitly hold that copyright law protects computer programming languages against unauthorized use. The issue has arisen, somewhat indirectly, in several prior cases. Most of these decisions point in a direction opposite to *Paperback*—toward a rule that language elements are unprotectable under the copyright laws.

A. *Reiss v. National Quotation Bureau*

The earliest decision in the field is *Reiss v. National Quotation Bureau*, which is often wrongly cited as having held that languages are copyrightable. In *Reiss*, Judge Learned Hand opined that copyright extended to a book of 6000 five-letter arbitrarily coined "words." ("ABBRA" would be an example of such a word.) The words were intended for use as cable code, where the users would supply their own private meanings for the coined words. Apparently, the defendant reproduced the plaintiff's book for competitive purposes.

The *Reiss* case did not involve anything analogous to a computer programming language, however. Reiss' book provided no linguistic syntax or meaning. At most, the book contained a proposed vocabulary, which was without any assigned meaning. Certainly, a real language must have syntax, as well as vocabulary. In any event, the entire holding in *Reiss* concerned the

---

32. An "uninterpreted" language may have syntax and vocabulary, without any assigned meaning, but vocabulary without syntax meets no definition of a language. *See supra* note 13. The term "real language" used in the accompanying text refers to an interpreted language. In *Reiss*, Judge Hand referred to uninterpreted languages (not so designated) as "a kind of blank Esperanto," and suggested that they could be protected by copyright. 276 F. at 718. However, he did not provide any explanation nor cite any precedent to support his assertion. He also left unstated what public policy such copyrights would possibly further.

Judge Hand's comment is not only immaterial *obiter dictum*, but ill considered. If such a "blank Esperanto" later turned out to correspond, for example,
constitutional requirement found in the copyright clause authorizing Congress to enact copyright laws—that copyright protections may extend only to "writings of authors." Judge Hand considered that the requirement was satisfied in regard to Reiss' code book because Reiss had tangibly manifested something in a recording medium (the printed pages of the book). In his view, that was enough to meet the "writings" requirement even though Reiss' book lacked any expression communicating any idea of Reiss as an "author." Perhaps by implication, Judge Hand also considered Reiss to be an "author" for constitutional purposes, on the grounds that he had created whatever was manifested in the book.

One might seek to extrapolate from Reiss the proposition that, to a kind of propositional calculus expressing the unification of gravitational and electromagnetic fields, an interpretation of copyright giving a monopoly over the use of that blank Esperanto would probably do considerably more harm than good to the promotion of scientific progress. See Diamond v. Diehr, 450 U.S. 175, 185 (1981) (laws of nature excluded from patent protection); Gottschalk v. Benson, 409 U.S. 63, 67 (1972) (patents on abstract intellectual concepts and phenomena of nature not allowed because they would preempt basic tools of scientific progress).

33. U.S. CONST. art. I, § 8, cl. 8. Congress lacks power to protect nonwritings under this clause. See The Trade-Mark Cases, 100 U.S. 82, 97 (1879) (trademark legislation held unconstitutional because not directed to writings of authors); see also Feist Publications, Inc. v. Rural Tel. Serv. Co., 111 S. Ct. 1282, 1288 (1991) (similar discussion).

34. It may well be that whatever one writes is a "writing," but the word "author" carries with it an implication that the author's writing must be an "original intellectual conception" resulting from "intellectual production, ... thought, and conception." Feist, 111 S. Ct. at 1288 (quoting Burrow-Giles Lithographic Co. v. Sarony, 111 U.S. 53, 59-60 (1884)). Query: can something without meaning or intrinsic intellectual content meet the foregoing standard? In Sarony, the court held copyright protection for a photograph of Oscar Wilde against a constitutional challenge based on lack of authorship, because Sarony's carefully posed photograph of Wilde was a means "by which the ideas in the mind of the author [Sarony, the photographer] are given visible expression." Sarony, 111 U.S. at 58. On the other hand, in Feist, the court denied copyright protection for the names and addresses in a telephone directory, because the work lacked sufficient originality and creativity to be an original work of authorship. See also Toro Co. v. R & R Prods. Co., 787 F.2d 1208, 1212-13, 1216 (8th Cir. 1986) (part number system based on random numbers "lacks the requisite originality to sustain copyright protection"). Accordingly, it is not sound copyright law that anything that a person writes is a writing of an author. Reiss' book may have been a writing, but he may not have been its author because none existed.
if meaningless vocabulary can be copyrighted, surely vocabulary to which syntax and meaning are added—that is, the elements of a real programming language—are copyrightable. But Reiss is of very questionable precedential value in deciding whether copyright protects real languages. The whole focus of the Reiss opinion is on whether something must have meaning to be a writing. Reiss is simply an interpretation of the "writings" limitation of the U.S. Constitution. The opinion contains no discussion of copyright principles concerning what is idea and what is expression. Indeed, Judge Hand indicated that he considered such issues wholly immaterial to the resolution of the case. Yet, whether vocabulary, syntax, and meaning are idea or expression are fundamental copyright-in-language issues. Whether something is a writing for constitutional purposes has no bearing on whether it is idea or expression.

35. All that Judge Hand said about the doctrine distinguishing idea and expression was "Baker v. Selden is too foreign to the case at bar to deserve comment." Reiss, 276 F. at 719.

The absence of any discussion of idea and expression in Judge Hand's Reiss opinion cannot be his mere oversight, for he was probably the most elaborately discursive judicial commentator on that issue. See, e.g., Sheldon v. Metro-Goldwyn Pictures Corp., 81 F.2d 49 (2d Cir.), cert. denied, 298 U.S. 835 (1936); Nichols v. Universal Pictures Corp., 45 F.2d 119 (2d Cir. 1930), cert. denied, 282 U.S. 119 (1931). The absence of discussion confirms the conclusion that Reiss has nothing to do with the relation of idea and expression.

36. That Reiss' code book was a writing does not, of itself, tell us whether its code words were idea or expression nor does it tell us whether someone who copied it copied idea or expression. As will appear from subsequent discussion in text, whether the plaintiff's work is copyrightable is usually the wrong question. The right question is usually whether the copyright in the plaintiff's work extends to what the defendant took. Thus, one should not ask whether Reiss' code book was copyrightable as a writing, but whether the defendant's conduct (use of Reiss' book) was copyright infringement. That question may raise the further question whether what the defendant took or used was part of a writing of an author, or even whether it was protected by copyright because of its relation to a writing of an author. But that is no longer the simple question raised in Reiss—was the code book a writing?

Asking overly simple questions, such as the one asked in Reiss, cannot lead to useful answers. The writings question should not even be considered in analyzing whether copyright protects computer programming languages (as it should not have been asked in Reiss). There is always some writing of the plaintiff somewhere in the field, or there would not be a lawsuit. A copyright owner cannot sue for copyright infringement without first applying to register the claim of copyright on which infringement is to be based, and a claim of copy-
Several other decisions are sometimes cited along with Reiss. Right cannot be registered without depositing a tangible copy of the work. 17 U.S.C. §§ 410(a)-(b), 411(a) (1988); Washingtonian Publishing Co. v. Pearson, 306 U.S. 30 (1939). Without a writing to deposit, there will be no copyright suit. If there is a writing, that is the writing on which the plaintiff will base its copyright infringement suit.

In the case of computer programming languages, there will of necessity be a literary work on which the suit is based. The work will be a book or a computer program; both are writings. See, e.g., Apple Computer, Inc. v. Franklin Computer Corp., 714 F.2d 1240 (3d Cir. 1983), cert. dismissed, 464 U.S. 1033 (1984). The question may then be raised whether the writing is that of an author. A threshold issue is whether there was enough creativity and originality involved to sustain the conclusion that the writing is an original work of authorship. Almost surely, in the case of an original computer programming language, or even a new dialect of an existing one, there is enough, since minimal creativity suffices. See Feist Publications, Inc. v. Rural Tel. Serv. Co., 111 S. Ct. 1282, 1287 (1991).

A further argument may be raised that Congress has no power to protect systems and ideas, assuming that computer programming languages should be so classified, under the copyright laws. This argument can be rephrased to invoke several constitutional provisions. First, it may be argued that ideas and systems, like the “facts” considered in the Feist case, do not have “authors”; they only have “discoverers,” and discoverers are not authors according to Feist. Id. at 1296. Second, the copyright clause limits congressional authority to passing copyright laws that promote the progress of knowledge (science and useful arts). See Sony Corp. v. Universal City Studios, Inc., 464 U.S. 417, 429 (1984). Giving out monopolies over ideas would hinder, not promote, the progress of knowledge. That is why copyright law “encourages others to build freely upon the ideas and information conveyed by a work.” Feist, 111 S. Ct. at 1290. Third, one might try to construct a first amendment argument. See Lee v. Runge, 404 U.S. 887, 892 (1971) (Douglas, J., dissenting to denial of certiorari) (serious first amendment questions would be raised if Congress’ power over copyrights were construed to include power to grant monopolies over ideas).

But it is superfluous to make constitutional arguments when the statute itself denies copyright protection to ideas and systems. See 17 U.S.C. § 102(b) (1988); see also Harper & Row, Publishers, Inc. v. Nation Enters., 471 U.S. 539, 556 (1985) (dictum). Indeed, Feist called that “[t]he most fundamental axiom of copyright law.” Feist, 111 S. Ct. at 1287. There is no reason to think that the statute allows protection of some ideas and systems that the Constitution places in the public domain. Thus, Feist interpreted the copyright statute’s concept of originality to be the same as that of the Constitution and held the facts in the copyright owner’s telephone directory unprotected as a matter of statutory interpretation, not because the statute was unconstitutional.

These constitutional arguments thus remain academic unless Congress should take it into its head to repeal § 102(b), directly or by implication, and expressly extend copyright protection to computer programming languages, notwithstanding § 102(b). For example, consider a notional § 102(c), reading
as precedents supporting the copyrightability of languages. But they are largely inapposite\(^{37}\) or may even suggest that languages are unprotectable.

**B. Hartfield v. Peterson**

A later code book case, *Hartfield v. Peterson*,\(^{38}\) has more complex facts and more fully illustrates the copyright issues that such cases may raise. The author compiled a set of code phrases taken from preexisting bank codes, and in his book he gave his interpre-

"Notwithstanding section 102(b), copyright protection extends to original computer programming languages described or embodied in a work of authorship." But that notion is unrealistic; Congress will not be moved to do that.

That is not to say that the policies behind the constitutional arguments are irrelevant. But they are relevant only because they illuminate what congressional intent "must" have been. Thus, in *Feist*, the court went on at length about what the Constitution requires, and then interpreted the statute on the supposition that Congress intended to track the Constitution. Similarly, the court has concluded that patents on algorithms would hinder, rather than promote, the progress of technology. But the Supreme Court accepted that argument by way of interpreting the patent statute to discern Congress' intent in enacting it; it thus concluded that Congress, although silent on the matter, must have intended to exclude algorithms. Parker v. Flook, 437 U.S. 584, 589, 595-96 (1978); *see* Diamond v. Diehr, 450 U.S. 175, 185-86 (1981); Gottschalk v. Benson, 409 U.S. 63, 67 (1972).

The proper issue is statutory construction, not what the Constitution permits, except insofar as the latter illuminates what Congress "must" have intended. *Reiss* simply addresses the wrong question. It would have been better to ask whether Congress intended to protect books of meaningless five-letter words, not whether it had power to do so. Because *Reiss* asks and answers the wrong question, we cannot learn anything useful from it about computer programming languages. It would be better to turn to such questions as what (if anything) did Congress intend to do about computer programming languages, and whether what it intended to do was a good idea or a mistake that it should be persuaded to rectify.

37. In American Code Co. v. Bensinger, 282 F. 829, 833 (2d Cir. 1922), the court protected a listing of code words. The *Bensinger* court stated "[t]hat a list of [code] words . . . is copyrightable . . . must be conceded." *Id.* at 833. But the only authority it cited (curiously, not *Reiss*) was a decision holding that directories are copyrightable as compilations of data. *Id.* (citing Jeweler's Circular Publishing Co. v. Keystone Publishing Co., 281 F. 83 (1922), recently overruled by *Feist* Publications, Inc. v. Rural Tel. Serv. Co., 111 S. Ct. 1282 (1991)). *Bensinger* did not mention the idea/expression issue. *Bensinger* is in accord with *Reiss*, but adds nothing to it and contains even less legal analysis. Moreover, it rests on overruled precedent.

38. 91 F.2d 998 (2d Cir. 1937).
tation of the meanings that banks had assigned to the terms. In some cases the plaintiff apparently made errors in describing meanings for the code phrases. The defendant copied many of the same phrases into his competing code book, including the errors.

To the extent that the plaintiff stated proposed meanings for code phrases in his own phraseology, there was individual expression of idea, at least unless the meanings for some reason dictated the phraseology. The defendant had no excuse to copy that phraseology—for example, what good reason did he have to copy the phraseology of the errors in explanation? The defend-

39. Id. at 999.
40. Id. at 1000. The opinion does not explain what was copied and what was protected, but that can be surmised. Apparently, a typical previous bank cable code phrase would be something such as “ABBRA 20,000 CADAB 12 HOCUS” to mean that a letter of credit is opened in favor of the addressee, who may draw upon it by presentation within 12 months of a sight draft, a clean bill of lading, and an invoice not to exceed $20,000. The plaintiff codebook writer collected many such code phrases, placed them in alphabetical order, and listed their meanings as he understood them and in his own words. Id. at 999. In doing so, he occasionally made mistakes, perhaps such as saying a “clear” bill of lading instead of a “clean” one. The defendant apparently identically copied many of the first book’s code phrases and words of explanation, including some errors. Id. at 1000.

41. See Morrissey v. Procter & Gamble Co., 379 F.2d 675, 678 (1st Cir. 1967) (where idea dictates phraseology, there is no copyright in phraseology (expression)).

42. Some care is needed to avoid too sweeping a statement about copying errors. The same errors appearing in a defendant’s book as in a plaintiff’s book is usually persuasive evidence of copying, absent a good explanation. See, e.g., Official Aviation Guide Co. v. American Aviation Assocs., 150 F.2d 173 (7th Cir. 1945); Adventures in Good Eating v. Best Places to Eat, Inc., 131 F.2d 809 (7th Cir. 1942); College Entrance Book Co. v. Amsco Book Co., 119 F.2d 874 (2d Cir. 1941).

But there is no copyright in facts, rightly or wrongly stated. “The most fundamental axiom of copyright law is that ‘[n]o author may copyright the facts he narrates.’” Feist Publications, Inc. v. Rural Tel. Serv. Co., 111 S. Ct. 1282, 1287 (1991) (quoting Harper & Row, Publishers, Inc. v. Nation Enters., 471 U.S. 539, 556 (1985)). Someone who makes up a series of imagined incidents from the life of Wyatt Earp and puts them into a book with the representation that the tale is no myth, see Lake v. CBS, Inc., 140 F. Supp. 707 (S.D. Cal. 1956), or who claims to tell a real experience with extraterrestrial beings, see Oliver v. Saint Germaine Found., 41 F. Supp. 296 (S.D. Cal. 1941), cannot use copyright to prevent others from recounting the same stories (saving an idea/expression caveat). See also Hoehling v. Universal City Studios, Inc., 618 F.2d
ant in this case apparently did just the opposite of what the Court excused in Baker v. Selden; here, the defendant copied how the plaintiff expressed his explanation of the system, as well as the system itself. Under Baker, the defendant was free to copy the system, but was not free to copy the same words in which the plaintiff expressed his explanation of the system.

C. The Shorthand System Cases

Two decisions have involved systems for writing in shorthand, namely, transcription of spoken English by speedily written symbols representing particular syllables or commonly used words.

972 (2d Cir. 1980) (no copyright protection of theory that Hindenburg Zeppelin was destroyed as result of fanciful events).

This line of authority suggests that the choice of words used to express the incorrect interpretation of the code phrases should be important. The copyright is in the words of expression, not in the interpretation.

43. 101 U.S. 99 (1879).

44. Id. at 104-05. Another, less satisfactory formulation of the principle of the Hartfield case is that the defendant had the right to copy the preexisting code phrases and their meanings from the public domain, but not to copy them from the plaintiff's book. See Hartfield v. Peterson, 91 F.2d 998, 1000 (2d Cir. 1937). This rationale is less principled, because the defendant had the right to copy idea from the plaintiff's book (but not expression). That is what Baker did in the Baker case, and the Supreme Court approved that. Baker, 101 U.S. at 104-05. The same is true in every other idea/expression case where the court held that the copied subject matter was idea.

This type of legal analysis substitutes the morality of the grade school classroom for copyright principle. Barring acts of burglary or other gross violations of public order, whence the defendant took the subject matter is immaterial if it is unprotected idea. See generally Bonito Boats, Inc. v. Thunder Craft Boats, Inc., 109 S. Ct. 971 (1989) (public has right to copy material unprotected by patent or copyright). Accord Feist Publications, Inc. v. Rural Tel. Serv. Co., 111 S. Ct. 1282 (1991) (permissible to copy unprotected subject matter from copyrighted directory/compilation).

45. Brief English Sys. v. Owen, 48 F.2d 555 (2d Cir.), cert. denied, 283 U.S. 858 (1931); Griggs v. Perrin, 49 F. 15 (C.C.N.D.N.Y. 1892). Such shorthand systems illustrate ways of transcribing language other than our ordinary alphabetic system, which more or less associates one symbol with one phoneme or several similar phonemes. In a shorthand system, one symbol may represent the syllable "la" and another symbol may represent "t;" a third symbol may represent the word "the;" a fourth symbol may represent "Very truly yours." In Japanese, the same word or phrase (for example, "computer programming language") may be represented in any of three ways: by an ideogram (a sort of hieroglyph), by syllabic representation, or in Roman letters (Romanji). These differences do not affect the idea/expression issue.
In both cases, the defendant published a book that imitated and explained the same shorthand system that had earlier been described in the plaintiff's book. Neither defendant, however, had copied the particular expression of the plaintiff's earlier book. Therefore, under the rule stated in the *Baker* decision, there was no copyright infringement.  

To the extent that one considers a shorthand system a language, the shorthand system cases suggest that languages are unprotectable systems or ideas. But the shorthand system cases do not so much involve the particular vocabulary or syntax of a specialized language as they do a new and different symbolic way of representing preexisting language with preexisting vocabulary, syntax, and assigned meaning.

The *Paperback* case, in at least one respect (unmentioned by the court), parallels the fact pattern of the shorthand system cases. To provide an alternative to pressing the keys corresponding to the initial letters of the input-command words of 1-2-3, the defendants numbered the command words, so that keys corresponding to the numbers could instead be pressed to indicate particular command words. Thus, the defendants created an

---

46. *Owen*, 48 F.2d at 556; *Perrin*, 49 F. at 15.
47. Whether one considers shorthand systems to be languages would appear to depend on how one interprets or extrapolates from the definition discussed in *supra* note 13 and accompanying text. If one focuses on the shorthand symbols written on a piece of paper, appropriately grouped (at a sufficiently non-atomistic level), perhaps a shorthand system literally meets the definition of language.

But apart from its visual appearance, the shorthand system is not a new language. It borrows the vocabulary and other features of English or whatever language it represents. Therefore, one may question whether shorthand is a different language from English. To be sure, shorthand has different symbols from printed or cursive Roman-letter representation of English; moreover, there is not a simple relationship between two representations. Hence, there might appear to be two languages. But for the other reasons previously stated, the shorthand system is just a set of symbols that can be mapped onto English, a preexisting language, whose vocabulary, syntax, and meaning it automatically incorporates.

Since whatever is original with a shorthand system is simply the symbols, it can be regarded as the "kind of blank Esperanto" to which Judge Hand referred in *Reiss*. See *supra* note 32. But the Second Circuit's ruling in *Owen*, see *supra* notes 45-46 and accompanying text, is inconsistent with Judge Hand's comments a decade earlier in *Reiss* about blank Esperanto, and may thus be considered to overrule them *sub silentio*.

alternative symbolic representation of the input-command language of 1-2-3, a kind of "1-2-3 shorthand." Should it be copyright infringement if another person copies the 1-2-3 shorthand? The shorthand system cases suggest that no particular system of assignment of numbers to command words could, in itself, be protected with a copyright.\(^4\)

D. \textit{Ashton-Tate Corp. v. Ross}

A recent Ninth Circuit decision addressed the copyrightability of a programming language similar to that involved in \textit{Paperback}. The logic of the court's opinion is that the vocabulary of an input-command language is idea rather than expression. But the court did not refer to languages, as such, in its opinion.

In \textit{Ashton-Tate Corp. v. Ross},\(^5\) Ross sought to base a claim of joint authorship on the fact that he provided a list of user commands to be embodied in the input-command language of the user interface of a computer program.\(^6\) Ross did not write any

\(^{(D. \text{Mass. } 1990).}\) For example, a set of input-command words any one of which could be selected from the first menu bar to start a command statement would be "3 Range, 4 Copy, 5 Move, 6 File . . . ." The user could press the keystrokes 3 or R to indicate Range, 4 or C to indicate Copy, 5 or M to indicate Move, and so on. The numbers are an alternative to the initial capital letters, as the hooks and curves of a shorthand system are alternatives to writing in Roman letters.

The \textit{Paperback} court did not discuss this input-command expedient or whether it could be protected by copyright. (No copyright appears to have been asserted in it and no one appears to have copied it.)

49. If called to rule upon it, the \textit{Paperback} court might well have considered the assignment of numbers to input-command functions to be a protectable nonliteral aspect of an underlying computer program that enabled performance of the functions. That would be consistent with the court's ruling about 1-2-3's command words. Probably, everything that could be said of the command structure of 1-2-3 could be said of another interface using numbers to designate input-command functions. For example, just as /RF and /FR are meaningful parts of a 1-2-3 statement and /RR and /FF are gibberish, see supra note 20, it might be said that /23 and /32 are meaningful parts of a statement in Paperback's system, while /22 and /33 are gibberish in that system. Similarly, there is about as much (or as little) creativity and expressiveness in Paperback's deciding that 7 shall represent the Printing function as in Lotus' deciding that P shall do so (or that the word "Print" shall do so).

50. 916 F.2d 516 (9th Cir. 1990).

51. \textit{Id.} at 520. The court's opinion does not disclose whether Ross provided syntax rules for the commands, and we may therefore assume that Ross did not. Presumably, he provided meanings to be assigned the commands, or they were
of the code of the underlying computer program—the engine of Ashton-Tate's "Full Impact" spreadsheet program.

Had the Ninth Circuit accepted Ross' claim to status as a joint author, he would have been entitled to a part interest in the copyright in the program. But the court rejected his claim and held his activity not to be authorship. The Ninth Circuit said that a putative joint author, to have actual "author" status, must contribute copyrightable subject matter (a copyrightable work) to the work of which he claims to be a joint author.\(^5\) That is, he must contribute protectable expression rather than merely unprotectable idea. Ross failed to meet that standard, the court held, because his list of words was a contribution of only idea, not expression.\(^5\)

The Ninth Circuit did not provide a detailed explanation of why Ross' list of input commands was an idea rather than an expression. It merely stated that Ross' list "simply does not qualify for copyright protection."\(^5\) Apparently, the court thought it obvious, and therefore unnecessary to explain, why a list of user input commands is an idea instead of expression. The Ross opinion did not mention copyrightability of programming languages as such; the opinion, however, is an important precedent because a major part of software activity in the United States occurs within the Ninth Circuit.\(^5\)

---

52. *Ashton-Tate Corp.*, 916 F.2d at 521.
53. Id. at 521-22.
54. Id. at 522.
55. The Ninth Circuit reached a similar result in S.O.S., Inc. v. Payday, Inc., 886 F.2d 1081 (9th Cir. 1989). In that case, the non-author created either a vocabulary list or merely a description of the tasks to be carried out by the computer program. The court held it not to be authorship activity. Id. at 1087. *Accord* Whelan Assocs., v. Jaslow Dental Laboratory, 609 F. Supp. 1307, 1318-19 (E.D. Pa. 1985), aff'd, 797 F.2d 1222 (3d Cir. 1986), cert. denied, 479 U.S. 1031 (1987).

The decision in Digital Communications Assoc. v. Softklone Distrib. Corp., 659 F. Supp. 449 (N.D. Ga. 1987), is sometimes cited as holding that a compilation of command words is protectable under a compilation copyright. That is

---

The Ross court held a computer programming language vocabulary to be idea rather than expression for purposes of determining existence of joint authorship. There is no reason why the same facts would not lead the same court to the same conclusion in a case in which it was necessary to resolve the idea/expression question for purposes of determining whether copyright infringement had occurred. There is no logical reason why vocabulary should be idea for determining authorship and expression for determining infringement, unless there is some significant difference in the interests at stake or in other relevant policies. Yet, in both cases the basic issue is the same: is a first party entitled to receive compensation from a second party when the first party creates, and the second party appropriates the vocabulary (and perhaps also the assigned meaning) of an input-command language? There is no apparent difference in the competing interests or policies at stake. Accordingly, there is no reason to have different criteria for distinguishing idea from expression in the two contexts. Ross therefore cannot be understood as a decision limited merely to criteria for joint authorship, as contrasted with the copyrightability of language elements.

E. NEC Corp. v. Intel Corp.

One other case deserves mention mainly because of what it fails to discuss and what was not contended. In NEC Corp. v. Intel Corp.,\textsuperscript{56} the issue was whether NEC had infringed Intel's too broad a reading of the opinion, however, for the court made it clear that it found infringement because the defendant had copied many features of the plaintiff’s screen display (menu) that were, in the court’s thinking, expression rather than idea. The copied features included the arrangement of command terms and parameters on the screen, the manner of “highlighting” on the screen, which keystrokes should be entered to select or designate a term or parameter, and certain abbreviations (such as “RQuest” for “Request”). The Softklone opinion appears to assume (indeed, suggest) that the defendant could have used the same command terms and parameters if it had arranged them on the screen differently and identified the keystrokes differently. Properly read, Softklone is consistent with Ross and S.O.S. Moreover, Feist Publications, Inc. v. Rural Tel. Serv. Co., 111 S. Ct. 1282 (1991), indicates that such a reading is required by the law.

\textsuperscript{56} 10 U.S.P.Q.2d (BNA) 1177 (N.D. Cal. 1989), prior proceeding 645 F. Supp. 590 (N.D. Cal. 1986) (opinion subsequently withdrawn because of recusal).
copyright in its microcode for microprocessor chips. Microprocessor chips are capable of executing a set of so-called "macroinstructions,"\(^57\) which in the aggregate comprise an "instruction set."\(^58\) Different microprocessor chips have different instruction sets. A company wishing to compete by selling a fungible substitute for another company's microprocessor chip will have to design its own chip to carry out the same instruction set as the other company's chip does. The microcode is a set of short programs that carry out operations (microinstructions) within the chip to execute the instructions of the instruction set. The instruction set of a microprocessor chip, in effect, defines a programming language that the chip executes.

In the *NEC-Intel* case, the instruction set defined the 8086 assembly language, a type of low-level source code. NEC's chip used the same 8086 instruction set as Intel's chip did, but the microcode that NEC used for the individual microinstructions was somewhat different. Intel contended, unsuccessfully, that NEC's microcode was substantially similar to, and therefore infringed, Intel's microcode.\(^59\)

Intel did not contend, however, that NEC had no right to make and sell a microprocessor chip using the same 8086 instruction set as did Intel's chip.\(^60\) The court's opinion did not state why Intel conceded the point. It is possible that Intel's 8086 instruction set was not original. More likely, however, Intel simply

---

57. A chip macroinstruction is an assembly language expression such as MOVE (a,b), which might mean "Move the contents of Register A to Register B." See IEEE *STANDARD DICTIONARY OF ELECTRICAL AND ELECTRONIC TERMS* 513 (3d ed. 1984).

58. An instruction set is the repertory of instructions that the chip can execute or carry out. The IEEE *STANDARD DICTIONARY OF ELECTRICAL AND ELECTRONIC TERMS* 445 (3d ed. 1984) defines "instruction set" as "[t]he set of instructions of a computer, of a programming language, or of the programming languages in a programming system." An instruction set implicitly defines vocabulary, syntax, and meaning for the instructions that it comprises. Related documentation explicitly defines such vocabulary, syntax, and meaning. For a more complete discussion, see D. Siewiorek, C. Bell & A. Newell, *COMPUTER STRUCTURES* 13-14 (1982).


60. Intel and NEC stipulated that Intel "has not claimed in this action that [NEC]'s use of such macroinstruction set as such violates any rights of [Intel]." *Id.* at 592.
shared a general assumption in the industry that the instruction set was not protectable by copyright. By the same token, that would be an assumption that 8086 assembly language is not protectable by copyright.

F. Comparison of Decisions in Field

It is thus seen that there are a number of precedents that are relevant to copyright protection of languages. But as yet no judicial decision explicitly addresses copyright protection of computer programming languages. The Reiss, Ross, and Paperback opinions may be arrayed, in that order, along a continuum of successively increasing relevance to the copyright-in-language issue.

The district court's opinion in the Paperback case is closest in point. The Paperback court opined in dicta that it saw no reason why a computer programming language could not be protected by copyright. The court then went on to hold the defendants liable for infringing the plaintiff's right to exclusive use of an input-command language (although not expressly). The input-command language involved in the Paperback case had all three elements of a computer programming language: vocabulary, syntax, and meaning.

The Ross case involved an input-command language vocabulary, which apparently had assigned meaning, but probably did not have syntax. The Ninth Circuit held in Ross that the vocabulary in question was part of idea rather than part of expression. Perhaps, the Ross case may be distinguished from Paperback on the ground that Ross involved only the vocabulary and associated assigned meaning of a command language—two out of the three elements of a computer programming language, while Paperback involved vocabulary, meaning, and syntax—three out of the three elements. One must then ask whether the Ross court would have reached the opposite result if Ross had also contributed syntax. Such a conclusion seems doubtful. Syntax, i.e., the rules for combining vocabulary into a well-ordered statement, is the most ab-

62. Ashton-Tate Corp. v. Ross, 916 F.2d 516, 521-22 (9th Cir. 1990).
stract of the three elements of a language. It is thus unlikely that something so abstract could impel some aspect of a work (here, a computer program) over the borderline from abstract idea to concrete expression. Ross and Paperback should simply be regarded as taking contrary views of idea and expression in computer programming languages.

Reiss held that a book of totally meaningless words (a vocabulary divorced from assigned meaning) was a writing protected against copying by a competitor. For reasons discussed earlier, the Reiss decision is not very relevant to determining whether a computer programming language should be regarded as part of idea or part of expression for purposes of copyright analysis.63 The Reiss court expressly refrained from considering the idea/ expression issue, stating that Baker v. Selden had no bearing on the issues.64

The shorthand system cases do not readily fit into this continuum, because they involve whether copyright will protect a particular new and different symbolic way of representing a given language rather than whether copyright will protect a new vocabulary and syntax of a new language. But because the shorthand system cases apply the rule of the Baker case to a system for representing language, they suggest the unprotectability of such systems under copyright law. To the extent that they are relevant, the shorthand system cases are in accord with Ross.

IV. REFINING THE QUESTION

The decisions discussed above may suggest that different considerations apply to reproduction of a computer programming language in the course of different uses. Reproduction of a language used in writing a new computer program may raise different issues from reproduction of a book and selling the copy as a competing book. Thus, Baker had no privilege to copy and sell the explanatory text of Selden's copyrighted book. Baker's privilege of use extended only to Selden's ideas and to so much of

63. See supra notes 32-36 and accompanying text.
Selden's expression as was necessary to use of the ideas explained in Selden's book.

Hence, one should not simply ask whether computer programming languages are copyrightable. A better question is: In what circumstances should an unauthorized reproduction or other unauthorized use be held a copyright infringement? It is not enough even to ask whether computer languages are protected by copyright. One must frame the question more narrowly to get sensible answers.

First, one must ask, "Protected by what copyright?" Several different copyrights may be relevant. There may be a copyright in a book describing the language. There may be a copyright in a compiler or interpreter program for translating the language from source code\(^5\) into machine language.\(^6\) There may be a copyright in a computer program using the language. The respective answers to whether copyright protects the language may be different for copyrights in such different kinds of work.

The next question for analysis is, "Protected against what acts?" Consideration of different acts may lead to different answers. Copying the code book of the Reiss case and selling the copy in competition against Reiss may be altogether different from sending a cable containing code words taken from Reiss' book. By the same token, making and selling a substantial copy of a computer program embodying a new language may have different legal consequences than writing and selling a wholly different program that uses that language.

A. Languages Taught by Books and Compilers

Consider the copyright in a book explaining a new computer programming language. The clearest case involves Kernighan


\(^6\) Machine language, or object code, is a low-level language directly executable in a computer. Id. at 1243, 1248. A compiler translates source code into object code. Id. at 1243.
and Ritchie's book first describing the C language. To be sure, it is copyright infringement simply to reproduce their book within a competing book. Even in a directly competitive book, however, it is clearly permissible to set forth so much of the material in their original book as is necessary to criticize or teach what the original book said, for that is "fair use." Most important, the copyright in the book does not create a monopoly over the art taught or described in the book. As Baker v. Selden teaches, a book on the art of making watches or medicines creates no monopoly over the practice of those arts in accordance with the book's teachings. Rather, publication of the book dedicates the art to the public. A book on writing computer programs is no different. Any book on any useful art must equally dedicate the art to the public.

The dedication of the art that the book explains is constructive, not necessarily volitional. It is by action of law, and arises from considerations of policy and social utility. The primary reason for the dedication is that monopolies over useful arts—industrial property rights—are ordinarily granted only upon the would-be monopolist's compliance with the requirements of an industrial property regulatory system; and the monopoly granted is subject to conditions and limitations that the industrial property system provides, not the possibly milder ones of copyright law.


68. The judge-made doctrine of fair use, now codified in part in 17 U.S.C. § 107 (1988), lists various uses as "fair" and therefore not to be held copyright infringement. These uses include "criticism, comment, . . . teaching . . ., scholarship, . . . research." The shorthand system cases, see supra note 45, may be viewed as excusing fair uses of the plaintiffs' books or as holdings that the copyrights in the books did not cover the shorthand systems as such. The opinions in those cases suggest the latter.

69. Generally, patent and other industrial property systems require a substantial technological advance over any prior work of others in the field; that is a condition of legal protection. The issue of advance is usually investigated by an expert agency before the industrial property right attaches. In contrast, a copyright system generally requires only originality, i.e., independent creation as contrasted with mere plagiarism, and there is no agency investigation before protection attaches. The point is discussed in Baker v. Selden, 101 U.S. 99, 102 (1879), and Burrow-Giles Lithographic Co. v. Sarony, 111 U.S. 53, 54 (1884).
Consider the copyright in a compiler or interpreter program (for example, such programs as "GWBasic," "QuickBasic," "Turbo Pascal," "Turbo C," and "Microsoft C," all of which are later-coined dialects based on the original BASIC, Pascal, and C computer programming languages). These works clearly raise an issue not addressed in the preceding discussion regarding an original book teaching a new programming language: what is the supposed act of copyright infringement? It is an established principle of copyright law that mere use is not copyright infringement. In the present context, copyright infringement occurs only when there is reproduction of a copy of the protected computer program work, distribution of a reproduced copy, or preparation of a derivative work based on or incorporating parts of the original work.70

Customers use a compiler or interpreter program for two main purposes. One purpose is to learn how to write computer programs in the particular language of the compiler or interpreter. That use of a compiler program is indistinguishable from the use of a book by reading and studying it, as one might study Selden's book on bookkeeping or the Kernighan and Ritchie book on C, in order to learn what the book teaches. Such a case is thus indistinguishable from Baker, and must be similarly resolved.

A second way that a customer uses a compiler (or interpreter) is in preparing another work. The user compiles (or interprets) a computer program that the user wrote. A programmer typically writes a computer program in a form of programming language known as source code. Before the computer program can be utilized in (executed by) a particular chip or machine, the program

---

For a discussion in the specific context of computer programs, see Karjala, *United States Adherence to the Berne Convention and Copyright Protection of Information-Based Technologies*, 28 Jurimetrics J. 147 (1988).

70 See 17 U.S.C. § 106(1)-(3) (1988). Other uses are generally beyond the reach of the statute. For example, it is not infringement to read a copyrighted book. See Foreign & Domestic Music Corp. v. Licht, 196 F.2d 627 (2d Cir. 1952); see also Twentieth Century Music Corp. v. Aiken, 442 U.S. 151, 155 (1975). Accordingly, someone who, without the consent of the copyright owner, engages in a use not within § 106's enumeration does not commit copyright infringement. Twentieth Century Music Corp. v. Aiken, 442 U.S. 151, 155 (1975); Fortnightly Corp. v. United Artists Television, Inc., 392 U.S. 390, 395 (1968).
must be translated from source code into machine-readable code (object code). The translation from source code to object code is effected by a compiler program (or interpreter program) designed for that source language and for that chip or machine.  

The object-code work uses words found (or quoted) in the compiler-program work, such as ADD, MOVE, PRINT, or the like. But the words will be strung together in the object-code work quite differently from any sequence found in the compiler program, even though the words are strung together in accordance with syntax rules taught or otherwise embodied in the compiler work. This kind of use of the compiler work is not a literal reproduction of a copy, assuming that the object-code work incorporates no modules or passages taken from the compiler program.

For the same reasons, the sale of the compiled object-code program is not the same as the distribution of a literal copy of the compiler work, because no such copy has been made. Again, for the same reasons, the object-code program is probably not a derivative work based on the compiler or incorporating it, assum-

71. An interpreter translates a source code into object code, one instruction at a time. Thus, the interpreter translates one instruction; the machine executes that instruction; then the interpreter translates the next instruction; the machine executes that instruction, and so on. A compiler translates the whole set of instructions of a source code into an object code, at one time. A compiled code executes faster, but use of an interpreter permits easier correction of errors and, at times, involves a smaller total body of code. See C. SIPPL, MICROCOMPUTER DICTIONARY 74-75, 196 (2d. ed. 1981) (definitions of "compiler," "compilers," "compiler vs. interpreter," "interpreter").

Similar considerations apply to programs known as "assemblers." A programmer may write a computer program targeted at a particular microprocessor or microcontroller chip, for example, the 80286 microprocessor used in "AT" personal computers. Such a program may be written in assembly language, a form of source code having a 1-1 correspondence between instructions in machine language (object code) and source code. An assembler program would be used to translate the source code into executable object code. See id. at 21-22 (definitions of "assemble," "assembler," "assembler language," "assembler program").

72. The proper concept of what is a derivative work is beyond the scope of this article. The statute defines a derivative work as a work that is based on a prior work and that recasts, transforms, or adapts the prior work—for example, a translation, dramatization, or condensation. 17 U.S.C. § 101 (1988). But those terms are not self-defining. The better view is that a later work may be held to be derivative work only if it borrows protected expression from the ear-
ing that the compiled object-code program takes no modules from the compiler (if it did, there would presumably be an infringing reproduction). It would therefore seem to follow that making and selling the compiled program violates no exclusive right based on the copyright in the compiler program.

An argument might be constructed, however, that the user reproduces nonliteral aspects of the work into a copy, with the result that a nonliteral copy is made. Or it might be said that the user bases the compiled program on nonliteral aspects of the compiled work, or incorporates such nonliteral aspects into the compiled work. If the argument were applied to the Kernighan and Ritchie book on C, rather than to a C compiler program, the argument would have to be rejected at once. The art of programming in C is a nonliteral element of the Kernighan and Ritchie book only to the extent that the art of Seldenian bookkeeping is a nonliteral element of Selden’s book. The art of programming in the C language can be a protected nonliteral element of their book only by overruling the doctrine of Baker v. Selden. That doctrine is part of worldwide copyright jurisprudence, and it certainly is fundamental to United States copyright law. The case should be no different in principle when the Baker argument is applied to a C compiler program, or one for any other computer programming language, than when it is applied to a book teaching that language.

A charge of copyright infringement might also be brought against a competitive compiler program that followed the same system of syntax and vocabulary usages that an original compiler program did, but was written in different code. This is a comp-
mon occurrence; there are many competing compilers on the market for widely used programming languages such as C and C++. Arguably, the initial Turbo Pascal compiler program did not teach the art of programming in the Turbo Pascal dialect of Pascal. Rather, it might be said, the compiler program was merely a noncommunicative tool for performing such programming, and not a work of instruction like Selden's book. Hence, the argument would be that the vocabulary, syntax, and meaning of Turbo Pascal language are protected nonliteral elements of the original Turbo Pascal compiler program work; and that, therefore, a subsequent Turbo Pascal compiler vendor cannot appropriate those elements from the earlier compiler work without incurring the guilt of piracy.

The better view, however, would reject that argument. The doctrine of *Baker v. Selden* is not limited to books; it pervasively defines the proper scope of protection that copyright law should accord any work of authorship. Certainly, the codification of *Baker* into section 102(b) of the copyright statute is not limited to the teachings of books. Instead, it is directed to any "work of authorship" and to "any idea, procedure, process, system, method of operation, concept, principle, or discovery" that the work has "described, explained, illustrated, or embodied." Programming in Turbo Pascal may properly be regarded as a procedure, process, or method of operation; any of these terms is a logically valid characterization. Further, the procedure is "embodied" in the compiler program to whatever extent that it could be claimed that the procedure is a nonliteral aspect or element of the program.

It follows, therefore, that copyright protection does not extend to programming in Turbo Pascal itself. By the same token it is not copyright infringement to engage in that conduct, or to embody that procedure into a subsequent compiler work. Invocation of phrases such as "nonliteral" does not change the conclusion that an original computer programming language first published in a copyrighted compiler program, or in any other

---

74. 17 U.S.C. § 102(b) (1988) provides: "In no case does copyright protection for an original work of authorship extend to any idea, procedure, process, system, method of operation . . . ."
work, should be regarded as part of the unprotected idea of that work, rather than as part of protected expression. The reason is clearly stated in Baker: copyright law is simply not intended to protect the art or procedure that a copyrighted work describes, explains, or does.\textsuperscript{75}

Further, considerations of equity and kindred doctrines support the conclusion that the copyright in a work describing or embodying a programming language confers no monopoly over the commercial use of the language. By publishing and selling a text, its author invites the public to learn what the text teaches.\textsuperscript{76} When a compiler and its documentation is sold, there is an implicit public invitation to learn to write programs in the language described in the documentation and embodied in the compiler. In the case of compilers advertised for use with personal computers, the invitation is explicit, as any reader of magazines carrying compiler advertisements will have observed.\textsuperscript{77} Having invited that use, the author/publisher is estopped by equity from charging purchasers with copyright infringement when they accept the invitation and write computer programs using the language. Even when the invitation is not explicit, existing legal doctrines such as implied license support finding an implied or constructive invitation.\textsuperscript{78}

\textsuperscript{75} Baker, 101 U.S. at 102.

\textsuperscript{76} Thus, in Baker, the Supreme Court said: "The very object of publishing a book...is to communicate to the world the useful knowledge that it contains. But this object would be frustrated if the knowledge could not be used without incurring the guilt of piracy of the book." Id. at 103. Although the Court purported to speak of the author's "object" as if a subjective fact inferred from the circumstances (vide "frustrating" the object), one may well conclude that the object is constructive rather than subjective, and that the Court would have disregarded any disclaimer by the author of intent to communicate knowledge to the world and thereby dedicate it to the public.

\textsuperscript{77} See, e.g., PC WORLD, Aug. 1989, at 78 (advertisement for QuickPascal Compiler); BYTE, May 1988, at 184 (advertisement for Wordcraft C package).

\textsuperscript{78} On implied license in patent law, see Aro Mfg. Co. v. Convertible Top Replacement Co., 377 U.S. 476, 484, 497 (1964); United States v. Univis Lens Co., 316 U.S. 241, 249-51 (1942); Met-Coil Sys. Corp. v. Korners Unlimited, 803 F.2d 684, 685-86, 687 (Fed. Cir. 1986). In Red Baron-Franklin Park, Inc. v. Taito Corp., 883 F.2d 275 (4th Cir. 1989), cert. denied, 110 S. Ct. 869 (1990), the court refused to apply the patent law's "implied use" doctrine to copyright. The court held that the sale of copyrighted video game equipment gave the purchaser no implied right to use the equipment for its intended purpose. Con-
B. Languages of Other Computer Programs

Next, consider the copyright in a more general computer program (for example, an applications program rather than a compiler) written in a new computer programming language. Does the copyright extend to the language, so that someone who ascertains the language by study of the first computer program (reverse engineering of a sort), and then uses the language to write other computer programs, infringes the copyright in the first computer program? Again, this would have to be a case of nonliteral copying, if anything. This is, in effect, a variation on the Baker case in which a notional, undidactic Selden lets others ascertain how he does bookkeeping without his explaining how or


A court could also hold that any attempt to withhold the right of use amounts to an impermissible derogation from the seller's grant. See British Leyland Motor Corp. v. Armstrong Patents Co., [1986] 1 All E.R. 850, 863 (H.L.) (judgment of Lord Templeman) (copyright in components of car cannot be used to compel car purchasers to buy replacement parts from seller's designee). That is equivalent to a constructive license or license implied by law.

The decision in Wilbur-Ellis Co. v. Kuther, 377 U.S. 422 (1964), is in the same vein, although its rationale appears to blend the foregoing lines of authority and that of the doctrine that monopoly rights are exhausted by the patentee's first sale of the patented product. In Kuther the Court held that the purchaser of a patented machine, by virtue of the purchase and the patentee's sale, acquired an unlimited right of use extending even to uses not contemplated by the patentee (for example, changing the capacity of the machine). Id.

Except for British Leyland, the decisions supporting an implied right of use involve patents. But the differences between patent and copyright monopolies do not have any apparent significance in regard to the purchasers' interest in the use of what they have purchased. Moreover, these decisions reflect the inverse of a principle accepted in patent and copyright law alike—that one who knowingly sells a product suitable only for practicing the subject matter of a patent or copyright, and unsuited for any noninfringing use, is liable as a contributory infringer for purchasers' infringements. See, e.g., Sony Corp. v. Universal City Studios, Inc., 464 U.S. 417, 441 (1984) (copyright case); Aro Mfg., 377 U.S. at 488 (patent case). It would seem to follow that a copyright owner who sells a program good only for compiling Turbo Pascal is responsible for the ensuing use and would have no ground to complain when the program was so used. To be sure, the Fourth Circuit rejected that proposition in the Red Baron case, but the fact that Congress overruled that decision suggests strongly that Congress considers the proposition as better reflecting proper notions of how the copyright laws should operate.
why he does it that way. In the preceding case of a compiler, the new language is in plain sight. In this case, the new language is infrastructure and some work is needed to dig it out.

Stripping away all of Selden's expository zeal, however, does not alter the applicable legal principle. Selden's art is still unprotectable by copyright. Further analogies can be found in the more conventional subject matter of copyright. There is no copyright monopoly in a literary or musical form or genre. The underlying reasoning of copyright law is that it would hinder more than promote the advancement of literature, music, and fine arts to give monopolies in genres to their pioneers. A particular style or form of doing something within the subject matter of

79. A book pioneering the Spenserian sonnet, terza rima, or sprung rhythm cannot confer a monopoly over that form or expedient. Breaking the prosценium arch, ending a scene with a rhymed couplet, verbing nouns and nouncing verbs, the mille fleures/International Style, pointilism, cubism, the consonance anglais, gout français, classical sonata allegro, and twelve-tone style are all uncopyrightable. If the contrary were the law, belle lettres and beaux arts would be immeasurably poorer.

80. Decisions involving imitation of style are sparse. Moreover, such decisions usually involve claims that the defendant has imitated the plaintiff's style as part of a scheme to palm the defendant's goods or services off on the public as those of the plaintiff. See, e.g., Sinatra v. Goodyear Tire & Rubber Co., 435 F.2d 711 (9th Cir. 1970), cert. denied, 402 U.S. 906 (1971) (imitation of Nancy Sinatra's singing style in radio and television commercials, decided under state unfair competition law); Lahr v. Adell Chem. Co., 300 F.2d 256 (1st Cir. 1962) (television commercial in which cartoon duck imitated "peculiar style" of voice of Bert Lahr).

Miller v. Universal Pictures Co., 18 Misc. 2d 626, 188 N.Y.S.2d 386 (Sup. Ct. 1959), rev'd, 11 A.D.2d 47, 201 N.Y.S.2d 632 (1960), aff'd, 10 N.Y.2d 972, 180 N.E.2d 632 (1961), involved simulation of the "Glen Miller style," defined as "a clarinet carrying the melody in unison with a single tenor saxophone—each instrument playing the tune an octave apart" with other saxophones supplying divided harmonies. Id. at 628, 188 N.Y.S.2d at 389. The appellate courts held that there was no property interest in the Glen Miller style.

One of the few copyright decisions is Atari, Inc. v. North Am. Philips Consumer Elec. Corp., 672 F.2d 607 (7th Cir.), cert. denied, 459 U.S. 880 (1982). The copyright owner created a video game having a nonviolent style, as contrasted with the usual mayhem featured in video games. The court stated that the plaintiff was not entitled to copyright protection against the defendant's imitation of the nonviolent style, but the plaintiff was entitled to protection of the particular characters and images that it had used to create that stylistic impression. The court also refused to accord the plaintiff exclusive rights to the expedient of scoring a player's performance by counting the number of dots that the player-token image "ate." Id. at 617-18.
Consider some variations on the last case suggested by the decision in *Paperback*. The underlying 1-2-3 computer program (the "engine" that drives the spreadsheet when a user operates it—the actual lines of code found on a 1-2-3 diskette sold by the publisher) was itself written in a pre-existing programming language. However, the underlying 1-2-3 program in part enabled the use of the new, higher-level command language that 1-2-3’s authors apparently originated. Thus, the 1-2-3 macroinstruction quoted earlier in the text is not itself an instruction literally occurring in the 1-2-3 engine. At least, it does not occur as such. It "occurs" only implicitly: the quoted macro follows the rules that the documentation and screen displays of 1-2-3 instruct its users to follow to cause performance of tasks, and the 1-2-3 engine contains code that enables or causes the performance of the...
task that the quoted macro describes or states.\textsuperscript{83}

Consider, first, the sale of a book of "canned macros" for 1-2-3 or for another such task-specific applications program. The macroinstruction quoted earlier is an example of what such a book contains. That macroinstruction is a short program for doing a task with 1-2-3. Books and diskettes containing libraries of such macros are available for sale—not only for 1-2-3, but for other computer programs as well (such as WordPerfect and dBase) which have command languages analogous to that of 1-2-3. Typically, proprietors of computer programs to which such macros relate encourage the creation and sale of books and diskettes of macros, because they increase the utility of and demand for the underlying computer program without substituting for it in the marketplace.

But suppose that, contrary to fact, the proprietor of the 1-2-3 computer program objected to the sale of a book or diskette of macros written in the command language that originated with the proprietor's copyrighted computer program. Should he recover for copyright infringement? In principle, the case seems indistinguishable from that of someone who applied Selden's bookkeeping system to particular kinds of bookkeeping problems, and published a book describing such applications. Accordingly, the answer should be that a computer program proprietor objecting to a macro book or diskette has no cause of action.

Finally, consider a case even closer to the facts of the \textit{Paperback} case. This last hypothetical will illustrate the implications of \textit{Paperback} and its potential impact on existing business practices in the software industry. Suppose that a competing spreadsheet seller markets a hypothetical "4-5-6" program with a different engine.\textsuperscript{84} The engine of 4-5-6 is written to respond to the identical keystrokes used as input commands in 1-2-3, and

\textsuperscript{83} The engine also contains code for performing tasks that any other well formed 1-2-3 macro can describe. In this regard, 1-2-3 is typical of a whole class of task-specific applications programs, which have specialized input-command languages that may be termed nonliteral aspects of their respective copyrighted engines.

\textsuperscript{84} This hypothetical is close to the facts in Ashton-Tate Corp. v. Ross, 916 F.2d 516 (9th Cir. 1990). \textit{See supra} notes 50-55 and accompanying text.
the keystrokes have the same functional significance. For example, the sequence /FR is a file-retrieval command statement in both systems. Thus, a user who knows how to operate 1-2-3 (a notional "fluent speaker" of 1-2-3) can operate 4-5-6 in the same way that a touch typist types equally well on one QWERTY keyboard of one manufacturer's typewriter or on another QWERTY keyboard of another manufacturer's typewriter. Suppose also, to avoid confusing the language issue with other issues involved in the actual *Paperback* case, that 4-5-6 has no menus and no help screens. Hence, no screen display or other aspect of 4-5-6 that users directly perceive is a literal copy of 1-2-3; there is nothing at issue here except the command language.

Clearly, there is no literal infringement of any aspect of 1-2-3. There are several possible theories of nonliteral infringement, however. First, the seller of 4-5-6 has created a facility for its users to input 1-2-3 macros (meaning macros in the 1-2-3 language) and, we may assume, to store (meaning "reproduce in copies," to use the words of the copyright statute) new and old 1-2-3 macros in disks. It may be argued that the seller of 4-5-6 induces users of 4-5-6 to commit copyright infringement by causing them to write computer programs in 1-2-3 language when they use 4-5-6. Second, it may be argued that the engine of 4-5-6 itself nonliterally copies or reproduces 1-2-3 by containing code enabling users to command tasks by using the 1-2-3 input-command language. Whether that conduct is copyright infringement turns, of course, on whether the copyright in the 1-2-3 program (essentially, the 1-2-3 engine) protects the copyright owner against such conduct.

The same hypothetical case can be stated in terms of operating system software as well, for such programs also use command languages. Sellers of DOS shells and sellers of substitute oper-

85. A DOS shell (for example, "1Dirplus" or "PCShell") provides a more visual and supposedly more "user friendly" user interface than MS-DOS does (although MS-DOS version 4 now includes such a shell). The shell acts as a translator interposed between the user and the MS-DOS system, so that the user can talk "plain English" to the shell while the shell translates what the user says into regular MS-DOS, a language at a farther remove from plain English. For example, a PShell user wanting to find a file down at the end of a highly branched sub-directory trail might move a cursor along a screen display picture
ating systems ("MS-DOS competitors")\textsuperscript{86} use the same command languages as do the operating systems at which they are targeted. MS-DOS shells and MS-DOS competitors probably will not generate copyright infringement litigation. Their existence in the marketplace is not a significant competitive threat to MS-DOS. But the case may be different for a notional "McClone-DOS," which we may postulate as an alternative to the Apple Macintosh operating system. This is so because of Apple's historic marketing and copyright enforcement policy; the difference illustrates some possible consequences of extending copyright protection to computer programming languages.

Generally, the IBM-compatible segment of the personal computer market, associated with the MS-DOS operating system, has operated under a regime in which any manufacturer can purchase and resell the MS-DOS operating system. Hence, anyone who wants to make and sell IBM-compatibles ("IBM

of the directory "tree" to the desired file, thus highlighting the name of the file, and would then press the return key. The shell then brings up the file. Without the shell, the user might have to remember and enter a difficult MS-DOS expression such as:

```
C:\ACCTG \RCVBLS \JONES \OVERDUE \FY90
```

The shell sends the signals corresponding to those keystrokes to the processing unit when, once the cursor is in the correct place on the screen, the user presses the return key. The shell inputs the necessary MS-DOS commands for the user to call up files, copy them, move them, rename them, and so on.

86. A shell leaves MS-DOS in place, and just tells MS-DOS what to do. But an alternative operating system program, such as Dr. DOS, replaces MS-DOS in the computer; Dr. DOS performs the same functions as MS-DOS does, but allegedly does so in a superior way. See Byte, Jan. 1991, at 277 (advertisement for Dr. DOS). Programs written in (or utilizing) MS-DOS command language operate with Dr. DOS, instead, without having to be rewritten. Thus the statement `COPY A:IDEA.EXP B:NONLIT.INF' will cause the file IDEA.EXP to be copied from Drive A to Drive B and renamed there as NONLIT.INF, whether the computer uses MS-DOS or Dr. DOS, even though the two operating system programs have different codes. Dr. DOS thus bears the same relationship to MS-DOS that the notional 4-5-6 bears to 1-2-3. Vocabulary, syntax, and meaning of the command language are the same, except insofar as Dr. DOS adds further language elements not found in MS-DOS. See Infoworld, Apr. 15, 1991, at 92 (advertisement for Dr. DOS).

There are programs such as "4DOS" which are intermediate in character and raise the same legal issues to be addressed. A program such as 4DOS replaces MS-DOS's COMMAND.COM program with an alternative or a superset, thus providing some enhancement. 4DOS uses a superset of the language used for MS-DOS.
clones") is free to do so. This has led to extreme price competition in the sale of IBM clones, and low prices. Arguably, it has also led to increased competition in terms of product features.

On the other hand, there are no Macintosh-compatible personal computers ("Macintosh clones") on the market at this time. As a result, Macintosh prices have been relatively well insulated from competition. Until now, nobody has been able to develop a noninfringing duplicate of the Macintosh operating system. Moreover, Apple is known for its zealous enforcement of its copyrights. Without a McClone-DOS, there can be no Macintosh clones. A McClone-DOS would pave the way for the kind of price competition in Macintosh clones that has long existed for IBM clones. A clone manufacturer could use the notional McClone operating system to market a McClone personal computer that would run existing software written for use in the brand-name Macintosh product, which is a necessary condition for clone success.

The notional McClone-DOS therefore poses the question

---

87. One marketing consultant estimated Apple's average margin at 52% and stated that "a clone company would be happy with a 45% margin." A Legal Mac Clone May Be on the Way, MARKETING COMPUTERS, Mar. 1991, at 17, 18.


An Apple official commenting on the possibility of Macintosh clones stated that Apple does not believe it possible to create a noninfringing clone, and "historically, we've been successful at aggressively defending our intellectual property rights." A Legal Mac Clone May Be on the Way, MARKETING COMPUTERS, Mar. 1991, at 17, 18.

89. In early 1991, Nutek Computers announced a set of three chips claimed to contain a noninfringing duplicate of the Macintosh operating system and to provide a graphical user interface dissimilar in appearance to Apple's. Nutek asserts that its chip set will offer manufacturers "a legal path to a working Apple Macintosh clone." Startup Claims Legal Mac-Clone Chip Set, Elec. Eng'g Times, Jan. 28, 1991, at 17, col. 1. Nutek's operating system presumably features the same command language as that of the Macintosh.
whether Apple would be successful if it sought to suppress sale of Macintosh-clone computers using McClone-DOS, by asserting copyright protection over the language to which McClone-DOS (and thus the Macintosh clone) responds. That would be an effective way to prevent any undesired competition in the sale of clone equipment, and surely a rational manufacturer would seriously consider it.

This is the point at which the Titanic hits the iceberg, by which I mean that the logic of Baker and the logic of Paperback collide. (What remains to be determined is which one is the iceberg and which is the Titanic.) The cases of the postulated 4-5-6 and McClone-DOS (and that of the DOS shells and MS-DOS competitors, as well) are no different in principle from the cases discussed earlier, and like them they are no different in principle from Baker. The 4-5-6 and McClone-DOS programs would therefore all appear not to infringe the copyrights in the programs whose command-language systems they use.

Yet the Paperback decision contradicts the logic of Baker. It must be anticipated that proponents of expansive copyright protection will frame arguments based on Paperback in favor of copyright protection of computer programming languages. Such arguments have already been made in copyright infringement litigation. 90 Sooner or later, there will be 4-5-6 and McClone-DOS cases. This article therefore turns to a critique of the reasoning of the Paperback decision, as applied to computer programming languages, and as that reasoning might apply to 4-5-6 and McClone-DOS.

The facts about 4-5-6 and McClone-DOS, as described above, are somewhat more simplified than those of the Paperback case. Paperback had extraneous facts that muddied the issues. For example, there may have been copying of the visual as well as the linguistic elements of the copyrighted program. But we may ap-

propriately follow the logic of the *Paperback* court's decision in cases challenging 4-5-6 and McClone-DOS purely on language grounds. Following that logic would require the conclusion that the command languages that the underlying 1-2-3 computer program engine and the Macintosh operating system program enable are parts of the protected expression, rather than the unprotected idea, of the underlying programs.

V. AN APPROACH TO PROGRAMMING LANGUAGES BASED ON *PAPERBACK*

The reasoning of *Paperback* may be paraphrased and restated in terms of the command-word vocabulary and syntax of 1-2-3 (or alternatively, that of the Macintosh operating system), explicitly recognized as a language, in the following terms:

The purpose, and therefore the idea, of such a command language is to permit a user to direct the performance of tasks by the applications program, here a spreadsheet program. Many different command languages could be devised that would accomplish that purpose, just as there now exist many computer programming languages—such as BASIC, Pascal, COBOL, FORTRAN, ADA, APL, PL/I, C, LISP, PROLOG—for writing ordinary computer programs.

That alternative ways exist to carry out, i.e., express, the idea of directing task performance in a spreadsheet, is a principal factor indicating that the command language of 1-2-3 is part of the expression, albeit a nonliteral part, rather than part of the idea of the underlying computer program.

In addition, the 1-2-3 command language is a particularly innovative, technically superior, and economically valuable aspect of the underlying computer program. It is important to reward and encourage such technological creativity.

For these reasons, the command language is protected expression, not unprotected idea. The defendant's unauthorized commercial use of that language in 4-5-6 is therefore copyright infringement.

A. *Idea and Expression in Languages*

The above logic is an incorrect interpretation of copyright law, of course. The reasoning is comprehensively flawed. The first flaw is addressing the issue of what is the idea and what is the
expression of a language, by considering the language itself as a work of authorship. A computer programming language is not a work of authorship. It is something that is embodied or used in one or another kind of work of authorship, such as in a book teaching the language or in a computer program using the language.

One can write an essay about the art of bookkeeping, as Selden did, or write a poem in the terza rima form, as Dante did. But that does not make the art of Seldenian bookkeeping or terza rima a work of authorship. The work is the essay or poem that has idea and expression. But the art explained, or device used, in the essay or poem does not have idea and expression. It thoroughly misstates copyright principles to attribute idea and expression to something not itself a work of authorship.

B. Purpose as Idea

Even if a programming language could have both idea and expression, the idea of the language would not be its purpose.91 There is no sound basis for identifying the purpose of a program-

---

91. This discussion assumes that one can plausibly determine the purpose of a programming language. But purpose is far from self-defining. There is inevitably a difficult problem in satisfactorily determining what is the purpose of anything. That is always an arbitrary process whose conclusion is thus questionable. The purpose of anything can be stated more generically and abstractly, or more specifically and concretely, as the speaker may desire. If that purpose is described more specifically, few or no alternative means will exist for accomplishing the stated purpose. Conversely, if the purpose is stated more abstractly, many conceivable alternatives will exist. Accordingly, there is a great risk that an arbitrary definition of the purpose at the outset of a legal analysis will determine the outcome that is built on inferences from the initial assumption. The result is therefore inevitably suspect, for verdict tends to follow sentence instead of preceding it.

In addition to the problem of bias in settling on an appropriate level of specificity, there is a further problem of vantage point. Poems, novels, computer programs, and other works of authorship do not have a purpose, in the usual sense of that term. Literally, none of them has any purpose, for purpose is an attribute of sentient, purposeful beings, not of inanimate works. Authors, publishers, and readers or users of works each have their own purposes, which may differ widely. Such a purpose may be analyzed or characterized in many ways, some highly subjective (for example, to make money or to edify the public) and others purporting to be more objective or based on a notional "reasonable man" standard. Even putting aside risks of disguised partisan bias, therefore, the precision with which purpose is determined is subject to the vagaries of vantage
ming language with idea when making an idea/expression analysis. The equation is fallacious and unsupported by copyright law. Moreover, the next step in the argument is also faulty. That step is the fallacious and unsupported proposition that if the purpose of a language or other work can be accomplished in several different ways, then the different ways are expressions of ideas, rather than ideas. Accordingly, an inquiry into the purpose of a programming language, and possible alternative means of effecting the purpose, will not support the stated conclusion that a language is an expressive aspect of, and thus protected by the copyright in, a computer program that enables the language.

The notion of equating the idea of a computer program with its supposed purpose or function, as well as the theory that if several ways of accomplishing such a purpose exist each of them is expression rather than idea, come from the Third Circuit’s opinions in *Whelan Associates, Inc. v. Jaslow Dental Laboratory, Inc.* and *Apple Computer, Inc. v. Franklin Computer Corp.* The *Whelan* court stated that for purposes of idea/expression analysis, the idea of the computer program involved in that case was to aid in the operation of a dental laboratory. The purpose, so described, could have been accomplished by structuring the computer program in several different ways. Yet, the defendant copied the way the plaintiff used. The court therefore found that the copied computer program structures were protected expression.

The *Whelan* court said that it was formulating a new rule applicable to the facts of the case before it. The proposition that the existence of several ways to accomplish a program’s purpose makes the different ways protectable expressions of ideas was derived from the earlier Third Circuit *Apple* opinion. In *Apple*, the

---

94. *Whelan*, 797 F.2d at 1236. “Where there are various means of achieving the desired purpose, then the particular means chosen is not necessary to the purpose; hence, there is expression, not idea.” Id. (footnote omitted). “Because that idea could be accomplished in a number of different ways . . . , the structure of the Dentalab program is part of the program’s expression, not its idea.” Id. at 1236 n.28.
court held that if other programs can be written that perform the same function as does a copyrighted program, then the copyrighted program is expression and not idea.95 This language was reflected in Whelan as the court attempted to explain its rationale for distinguishing expression from idea in computer program cases.

The two Third Circuit decisions drew on two strands of legal precedent. One strand was that if an idea can be expressed in several ways it is not infringement of a copyright in one such expression of the idea for a later comer to express the same idea in another way.96 The second strand was that of Baker and its progeny, particularly Morrissey v. Procter & Gamble Co.97 These decisions hold that when an idea is capable of being expressed only in a few ways, the expression stands on the same footing as the idea (the expression "merges into" the idea) and cannot be given copyright protection lest the effect be to give a monopoly of the idea.98 The Third Circuit's Apple-Whelan rule inverts the propositions stated in these precedents to reach the conclusion that when several ways exist to accomplish a purpose or function, the different ways are protectable expressions. As the following analysis shows, however, it is a logical fallacy to claim to derive the Apple-Whelan rule from the propositions stated in these two strands of precedent.

The first principle, that one way for expressing an idea does not infringe another way, is simply another way of saying that copyright protects expressions of ideas, but not ideas themselves. Hence, when two expressions of an idea are different from one

95. Apple, 714 F.2d. at 1252. "If other programs can be written... which perform the same function as an Apple's operating system program, then that program is an expression of the idea and hence copyrightable." Id.
96. See, e.g., Dymow v. Bolton, 11 F.2d 690, 691 (2d Cir. 1926).
97. 379 F.2d 675, 678 (1st Cir. 1967).
98. Some of these decisions, such as Morrissey, hold the work uncopyrightable when the idea can be expressed in only a very few ways. Id. at 678-79. Other decisions hold the work copyrightable, but not infringed when the only similarity is in respect of borrowing expression of ideas capable of only a limited number of ways of expression. See, e.g., Data East USA, Inc. v. Epyx, Inc., 862 F.2d 204, 208 (9th Cir. 1988); Atari, Inc. v. North Am. Philips Consumer Elecs. Corp., 672 F.2d 607, 616-17 (7th Cir.), cert. denied, 459 U.S. 880 (1982); Durham Indus., Inc. v. Tomy Corp., 630 F.2d 905, 912-13 (2d Cir. 1980).
another, neither one of them infringes any copyright in the other. There is a vast gap between that clearly valid proposition and the proposition sought to be inferred from it, which is that the existence of two or more ways of accomplishing something makes the several ways of accomplishing something expressions and makes the something accomplished an idea. The inference does not follow. Moreover, the several ways of accomplishing something can be ideas rather than expressions, as will appear below.

The *Apple, Whelan,* and *Paperback* courts correctly recognized that the idea/expression doctrine is an analytical method (a tool) for deciding whether a taking from a work is an infringement. But those courts misapplied that method when they applied idea/expression analysis to what a work *does* rather than to the pattern of the work itself. *Idea* is simply a term applied to any relatively abstract formulation of the elements of a work, or of a pattern that serves to describe those elements.9 The term *idea* is used in contradistinction to a highly specific formulation of the elements of a work or their pattern, to which the term *expression* is applied. An extreme paradigm of expression might be the verbatim text of a work—the actual lines of dialogue of a play, or the very words on the pages of a book. An extreme paradigm of idea would be the most general and abstract possible statement of what the work is about, such as "a true and tragical history of Richard II."100 An idea of a work is thus an abstract formulation of what the work is, not of what it is supposed to accomplish or reason it was created.

Works have many ideas and many expressions; a great number of different patterns of successive, differing generality characterize a work, as more or less detail is included in the characterization.101 Something can be an idea without being at the very extreme of generality and abstraction, and something can be an expression without being at the very extreme of concreteness.

99. See Nichols v. Universal Pictures Corp., 45 F.2d 119, 121 (2d Cir. 1930), *cert. denied,* 282 U.S. 119 (1931) ("Upon any work . . . a great number of patterns of increasing generality will fit"). For a discussion of the difficulties in applying the patterns test, see B. Kaplan, *An Unhurried View of Copyright* 48 (1966).

100. *Nichols,* 45 F.2d at 121.

101. See id.
Clearly, several different ways of doing something can each be ideas for carrying out another idea that is at a higher level of abstraction. For example, almost anyone will agree that recounting a story about a boy and girl who love one another but have difficulties in realizing happiness is idea, not expression. That idea can be particularized by adding further details that make the idea more specific. For example: that their respective parents are engaged in a feud, which stands as an obstacle to their marriage; that they decide to marry secretly; that the boy kills a relative of the girl; and that they enter into a suicide pact. None of those additional particularizations will convert the unprotected idea into protected expression. Each different way of particularizing the idea is still another idea, and is not yet concrete enough to be expression. The same idea can also be expressed in different literary or other media (for example, as a novel, poem, play, or video game). None of those different ways of telling the story is expression rather than idea.

There are many successive levels of idea that have to be traversed to move from the general idea of recounting a story about a boy and girl who love one another but have difficulties in realizing happiness, at one end of the idea/expression continuum, to the specific expression of Romeo and Juliet, at the other end. At each level of increased specificity, alternatives may be chosen. Nevertheless, as long as the pattern remains general enough, the fact that alternatives exist at that level of the pattern does not, of itself, demarcate the boundary between idea and expression. Whelan, Apple, and Paperback misconceive the idea/expression doctrine when they assert otherwise.

Trying to derive the Apple-Whelan principle by inverting the rule stated in Baker and amplified in the Morrissey case is another exercise in fallacious reasoning. The Baker-Morrissey rule may be expressed loosely as the following proposition:

IF a given function can be performed by only a few expedients, THEN the expedients are idea (rather than expression).

In the same manner, the Apple-Whelan rule may be expressed loosely as the following proposition (putting NOT into and thereby negating each clause of the preceding proposition):
IF it is NOT the case that a given function can be performed by only a few expedients, THEN the expedients are NOT idea (but are expression).

These statements can be represented, for the Baker-Morrissey rule as the proposition \( P \text{ IMPLIES } Q \), and for the Apple-Whelan rule as the proposition \( \text{NOT-} P \text{ IMPLIES NOT-} Q \). But it is simply incorrect to purport to derive the second proposition from the first. The logically correct inversion (contrapositive) of \( P \text{ IMPLIES } Q \) is \( \text{NOT-} Q \text{ IMPLIES NOT-} P \), rather than Apple-Whelan's \( \text{NOT-} P \text{ IMPLIES NOT-} Q \). By the same token, the correct inversion of the Baker-Morrissey rule is not the Apple-Whelan rule, but instead:

IF expedients are expression rather than idea, THEN the given function can be performed by more than only a few expedients.

C. Functional Aspects of Programming Languages

The argument based on Paperback for holding computer programming languages to be protected, expressive, nonliteral elements of computer programs in which they are found or embodied is comprehensively flawed. As a legal analysis, it will not withstand scrutiny. Moreover, even if its logical fallacies are put aside, the Apple-Whelan rule is unsound because it misunderstands the policy of the Baker decision.

Apple and Whelan turn the policy of Baker upside down. It does not matter that there are many ways to perform a function that a computer program carries out, or many ways to accomplish its purpose, if some ways (and, in particular, the way sought to be protected by a copyright) are faster, cheaper, or otherwise more satisfactory than other ways. To grant a first user of a superior way the exclusive right to use that way for some indefinite time (approximately seventy-five years) may relegate later would-be users to slower, more cumbersome, costlier, more error-prone, and otherwise less satisfactory ways. This makes computer-program copyrights the equivalent of a patent, without satisfying the stringent requirements of the patent process. The basic teaching of Baker is that copyright law must not be used to grant pseudo-
patents; only the patent system should give out patents.¹⁰² By the same token, extending the *Apple-Whelan* rule from computer programs to computer programming languages extends and amplifies the injury to policy which that rule works.

Programming languages, like other aspects of computer programming, may be functional. Some languages may be faster to code or compile, easier to debug or modify, more economical in memory usage, or less prone to errors in coding in particular applications. It is possible that any algorithm or function can be expressed in any language, if a programmer is skilled and persistent enough. But some languages make it easier than others to program particular algorithms or functions. For example, it is easier to implement a list processing procedure in LISP than in FORTRAN, but conversely easier to implement a complex mathematical computation in FORTRAN than in LISP.

¹⁰² The principle goes beyond copyrights and is more pervasive than *Baker*’s teachings. In *Sears, Roebuck & Co. v. Stiffel Co.*, 376 U.S. 225, 230-31 (1964), and *Compeo Corp. v. Day-Brite Lighting, Inc.*, 376 U.S. 234, 238-39 (1964), the Supreme Court held that state unfair competition law could not be used as a substitute for or complement to the patent system. In *Bonito Boats, Inc. v. Thunder Craft Boats, Inc.*, 489 U.S. 141 (1989), the Court held that a state “plug molding” law could not be used to hand out patent-like monopolies. These cases differ from *Baker* in that they rest on the Supremacy Clause and involve preemption of state law, rather than harmonization of different federal intellectual property laws as in *Baker*.

Other decisions interpreting federal intellectual property laws teach that non-patent laws must not be used to protect functional advances in technology in the manner that the patent laws do. *See In re Morton-Norwich Prods.*, Inc., 671 F.2d 1332, 1339-41 (C.C.P.A. 1982) (trademark law must not be used to protect functional and utilitarian features); *Barofsky v. General Elec. Co.*, 396 F.2d 340 (9th Cir. 1968), *cert. denied*, 393 U.S. 1031 (1969) (utilitarian features cannot be protected under design patent law but must rely on utility patent law).

It is fair to say that this body of law stands for the general proposition that patent-like monopolies must not be awarded except under the patent laws and in accordance with their safeguards and limitations. This principle, of course, has its limits. *See Kewanee Oil Co. v. Bicron Corp.*, 416 U.S. 470 (1974) (patent law does not preempt state trade secret law). The limits are probably defined by paramount or overbalancing state interests, and perhaps also by the existence of interests seemingly different in kind and applicable to a range of contexts much broader than industrial property. Decisions on copyright pre-emption of state law suggest such limits. *See, e.g.*, *Harper & Row, Publishers, Inc. v. Nation Enters.*, 501 F. Supp. 848, 852-54 (S.D.N.Y. 1980), *aff’d*, 723 F.2d 195 (2d Cir. 1983), *rev’d on other grounds*, 471 U.S. 539 (1985).
Although each procedure could be implemented into computer program code in either language, the implementation will be completed faster and with fewer coding errors by using the language that is more appropriate to the procedure.\textsuperscript{103}

That is why a programming language explained or used in a copyrightable work (such as a book or computer program) is, properly considered, what the \textit{Baker} decision calls an "art," and what section 102(b) of the present U.S. Copyright Act calls a "procedure, process, [or] system." Copyright law may protect the specific exposition of the art, procedure, process, or system—but not the art, procedure, process, or system itself. If that is to be done, it should be done under a scheme of industrial property law.

\section*{VI. Using Copyright to Encourage Technological Advance}

The \textit{Paperback} opinion asserted a further reason to consider languages protectable expression rather than unprotected systems or procedures. The \textit{Paperback} court contemplated an inquiry under which one would ask whether the subject matter in controversy has commercial value, whether it is socially desirable to encourage innovation and technological progress in the field of devising improvements in the subject matter, and whether rewards to creators (and investors in creation) will call forth a desired added increment of such innovation.\textsuperscript{104} Presumably, the answers to all of those questions would be affirmative. The court inferred from the results of that inquiry that courts should apply copyright protections to user interfaces by holding them protected expressions of the idea of some copyrighted work. In principle, one could reach the same conclusion for computer programming languages, as the earlier passage in text suggests.

\begin{footnotesize}
\begin{itemize}
\item \textsuperscript{103} See Pancake \& Bergmark, \textit{Do Parallel Languages Respond to the Needs of Scientific Programmers?}, \textsc{Computer}, Dec. 1990, at 13. "It is always possible to construct an accurate implementation [of an algorithm into code] using an inappropriate language. The transformation process is more tedious and error prone, however, when the conceptual models supported by the language relate only peripherally to the problem-solving model of the programmer." \textit{Id.} at 17.
\end{itemize}
\end{footnotesize}
That conclusion is untenable for several reasons. There appears to be a gap in the reasoning in that the correct inference from the premises would be, at most, that it might be wise for some proper authority to ordain that some sort of intellectual or industrial property right should apply to languages. Whether the right should be a copyright, patent, utility model, or some entirely new concept remains unresolved. Certainly, not every social need can or must be satisfied by means of copyright law.

A. Institutional Considerations

Even more important, the right institution for determining what sort of right, if any, should apply is Congress, not the judiciary. One need not question the wisdom of the ultimate conclusion that there should be some sort of intellectual property protection for computer programming languages. But surely an initial determination preceding creation of any such legal right should be a legislative decision that creating the right would do society more good than harm, i.e, the social benefit of the system would exceed the cost. Making that decision in the first instance is not an appropriate judicial function.

To be sure, a proponent of protection can always (and doubtless will) argue that Congress has already made a decision in favor of protection by passing a copyright law phrased in terms so general and ambiguous that they permit computer programs to be considered literary works akin to poems and novels. By the same token, those terms permit programming languages to be regarded as potentially protectable nonliteral aspects of such works.106

This argument, however, attributes to the legislature an intent


106. An extended general argument along these lines, but not specifically in regard to languages, may be found in A. CLAPES, SOFTWARE, COPYRIGHT, AND COMPETITION (1989), and in a predecessor work, Clapes, Lynch & Steinberg, Silicon Epics and Binary Bards: Determining the Proper Scope of Copy-
where none is evident. Moreover, it is at least equally arguable that Congress made a decision in favor of competition and free access to systems and other "ideas" by codifying Baker into the statute. Given the importance of access to programming languages (to both language users and the public), and also given the prior assumption that copyright did not protect languages because they are part of idea, explicit congressional legitimization of any copyright protection of languages appears necessary, at the very least, to preserve the appearance of a rule of law, rather than fiat.

In addition, the institutional procedures available to Congress (and unavailable to courts) suggest the superiority of the legislative process over the judicial process in devising appropriate forms of legal protection for programming languages. Congress can weigh a broader range of issues than adversarial litigation permits, and it can compromise or trade-off competing interests whereas courts generally refuse to do so. Both of these institutional factors are relevant to programming languages.

B. Interests of Users

The rights of users and the public are usually not considered in litigation between a plaintiff copyright proprietor and a competitor-defendant. They nonetheless deserve attention. The seller of a product embodying a language, such as the command language of 1-2-3 encourages the public to learn to speak that language fluently. Learning to effectively operate the 1-2-3 spreadsheet program, or any other program (such as a word processor, database manager, or an operating system), requires an end user to invest considerable time and effort in learning the input-command language. This training pays the end-user dividends in rapid and efficient use of the program.

The legislative hearing process may well result in a conclusion that it is unfair to end-users to condition their enjoyment of such benefits upon their remaining customers of the original vendor. The conclusion may be drawn that the value of learning to speak

fluently, for example, 1-2-3 should be the property right or relational interest of the end-user who made the investment, not of the vendor of 1-2-3. This example of the sort of conclusion that Congress may more readily or properly draw than courts highlights the lack of wisdom of recent judicial legislation in the field, as in the *Paperback* case.

A related public interest consideration involves de facto standardization resulting from widespread public education in and use of a particular programming language. Standardization may occur because of the language's inherent functional advantages, or simply by historical accident. A programming language may become so universal in its field that it is like the QWERTY keyboard, or the standard automobile H-gearshift arrangement. The language may lack intrinsic superiority over alternative languages, but externalities or extrinsic considerations may nonetheless dominate the marketplace. In such circumstances, a would-be market entrant may find that it is commercially impracticable to try to persuade users to adopt and learn a different programming language.

Hence, it seems clear that access to the language may be commercially indispensable. In such circumstances, the virtually unlimited availability of injunctions under copyright law may have such market-distorting effects that application of copyright pro-

---


108. In the dispute over the 1-2-3 language the defendant spreadsheet seller claimed that 1-2-3 had become a de facto standard in the spreadsheet field, because of its dominant sales position in the market. Suggestions to that effect have been made about the dBase language in the database manager field. Similar considerations may apply to more general programming languages. At one time, for example, BASIC may have enjoyed a de facto standard status. In scientific applications, FORTRAN at one time was (and may yet be) the de facto standard programming language. At this time, C probably has a de facto standard status for many types of programming application. In personal computer operating systems, DOS (PC-DOS and MS-DOS collectively) may also be a de facto standard.

It is doubtful that the same can be said of the Macintosh operating system. Equivalent considerations may nonetheless apply because of "lock in" of existing users. See Digidyne Corp. v. Data General Corp., 734 F.2d 1336 (9th Cir. 1984), cert. denied, 473 U.S. 908 (1985) (RDOS operating system held to effect "software lock-in" of users of Data General computers).
tions to the subject matter may be unwise policy; other copyright remedies may also be so overwhelming that they have market-distorting effects.\textsuperscript{109} Congress is free to weigh the effects of these remedies on software progress before deciding whether to enact legislation conferring copyright-like rights in languages.

The market-distorting effects of extending copyright remedies into this field could do more than deprive the public of the price benefits of additional competition. A further result could be to impose unwanted learning costs on the public and to impose other transaction costs on users. This can occur in the case of languages if an interpretation of copyright law coerces users to choose either to remain dependent upon the program that pioneered a proprietary language, or to learn a new language to enjoy the benefits of enhanced functionality offered only by the programs of later market entrants.

C. \textit{Compromise Solutions}

Congress is free to strike balances and make compromises that courts cannot. In intellectual property litigation, there is ordinarily a total winner and a total loser; Solomonic compromises are usually infeasible. If the court finds copyright infringement liability, the plaintiff is entitled to a permanent injunction and the defendant may be liable for a crushing monetary award. Alternatively, a plaintiff may take nothing at all, leaving defendant free to engage in the disputed practice (exploiting without limitation the results of the plaintiff's unprotected creativity). Polar ex-
tremes govern; intermediate solutions ordinarily cannot be adju-
dicated under patent and copyright law.

A rational balance of interests in the field of computer pro-
gramming languages may well call for a compromise and inter-
mediate solution. Congressional legislation is superior to judicial
legislation in this field because it is not necessary for Congress to
embrace either polar extreme. Congress may conclude that in-
dustrial property incentives are necessary or desirable to promote
programming language innovation, and thus promote technologi-
cal progress in the software industry. Congress would be free to
balance and trade off interests by enacting legislation calling for
reasonable royalties but denying injunctive relief for program-
ming language proprietors. The availability of that approach

110. Congress has provided this type of compromise option under the copy-
right laws to balance interests in musical compositions. See 17 U.S.C. § 115
when United States Government infringes copyright or patent; reasonable-roy-
alty compensation instead). Congress worked out a much more elaborate com-
promise in the case of semiconductor chip products. See Semiconductor Chip
compromise system for computer programming languages are beyond the scope
of this article. In summary, such a system could provide an industrial property
right under a registration, non-examination procedure having the following pos-
sible characteristics. Injunctions against unconsented-to use would not auto-
matically be available. The proprietor of rights would be entitled to reasonable
and entire compensation for use by others, but to injunctions only against per-
sons who unreasonably refused to compensate the proprietor for their use of the
subject matter. A condition of protection would be novelty and technical ad-
vance, to be determined in the event of litigation rather than as part of the
registration process. The duration of protection would be comparable to other
industrial property rights recognized under the Paris Convention for the Pro-
tection of Industrial Property, Mar. 20, 1883, 25 Stat. 1372, T.S. No. 379, as
would be considerably shorter than that of a copyright under the Berne Con-
Trademark rights, but not moral rights, would exist to prevent mislabelling of
extensions (supersets) and dialects. The proper test for infringement might
present some difficulties in the case of dialects; there might be a tendency not to
consider any but verbatim or near-verbatim duplication as infringement, to
avoid difficult questions such as whether Pascal infringed Algol, or ADA in-
fringed Pascal.

Detailed issues of that sort are inevitable whenever a system to protect indus-
trial property is devised. Clearly, expert deliberation over a long period, with
to courts under existing copyright law is very doubtful. 111

VII. CONCLUSION

Little in the nature of things governing U.S. intellectual property (including the Berne Convention) prevents Congress from determining that original programming languages shall be copyrightable, and that using them without authorization is copyright infringement. 112 By the same token, Congress can mandate the contrary rule.

many iterations of the product, is better calculated to produce sound results than is the adversarial/judicial process.

111. The Ninth Circuit has performed some interesting experiments in this field, but they do not provide a solution for the general case. In Universal City Studios, Inc. v. Sony Corp. of Am., 659 F.2d 963 (9th Cir. 1981), rev’d, 464 U.S. 417 (1984), the Ninth Circuit found a seller of videotape recorders liable for contributory infringement of the copyright in films shown on TV, but also found permanent injunctive relief inappropriate. The court instead awarded the equivalent of a continuing royalty on equipment sales. The Supreme Court reversed, holding that there was no infringement and thus no contributory infringement, mooting the remedy issues.

In Abend v. MCA, Inc., 863 F.2d 1465 (9th Cir. 1988), aff’d, 110 S. Ct. 1750 (1990), the Ninth Circuit attempted to adjust the equities between a copyright owner and a derivative-work licensee ousted from its license without fault on its part. The court awarded only monetary relief amounting to continuing royalties or a share of future profits. The Supreme Court mentioned in its opinion that it did not grant certiorari on the issue of remedy, and it did not address the merits of the remedy. 110 S. Ct. at 1757-58.

Thus, the Ninth Circuit has twice attempted to fashion a compromise verdict in cases where the equities and the issue of copyright infringement were close. Both decisions involved very unusual fact patterns, and other lower courts have been less innovative in tailoring relief to balance the equities of the case. Application of the Sony-Abend principle to computer programming languages, even in the Ninth Circuit, is unlikely. In addition, the United States has since become a signatory to the Berne Convention, which does not permit compulsory licensing. Arguably, denial of injunctive relief in favor of reasonable-royalty relief would violate the Berne Convention. See, e.g., Karjala, United States Adherence to the Berne Convention and Copyright Protection of Information-Based Technologies, 28 Jurimetrics J. 147, 150-51 (1988).

112. There are some possible constitutional limits. See supra note 36. But courts have been reluctant to second-guess Congress on how best to effectuate the constitutional purpose of encouraging the progress of knowledge, once Congress speaks. See Sony, 464 U.S. at 429-30. Except for the decision in The Trade-Mark Cases, 101 U.S. 82 (1879), which held trademark legislation unconstitutional because it was not directed to the writings of authors, no copyright law has been struck down.
The absence of any specific legislative statement on computer programming languages compels recourse to case law and the codification into the current copyright statute of the Baker doctrine. This inquiry indicates that probably even a novel, useful, innovative, original computer programming language should be held to be an idea rather than an expression, and thus unprotected by copyright.

A sensible answer to the issues as to whether programming languages are idea or expression protected by copyright law should be based on a clarification of the issues. The questions should be posed in terms of an idea aspect or an expression aspect of a particular kind of work, and in relation to the particular type of allegedly infringing conduct.

The answer is clearest in the case of a book written about a programming language, for then no state of facts would appear to permit copyright protection of the language taught in the book on the basis of the copyright in the book. The issue becomes more difficult when the question concerns copyright in other works embodying the new, original, highly creative language, such as compilers and task-specific programs, and when the challenged use is directly competitive with the copyrighted work. Even then, however, on balance the better view is that copyright protection should not extend to any use of programming languages. Such languages should be, in all material respects, on the same copyright law footing as was the bookkeeping system in Baker.

This is not to say that computer programming languages are undeserving of intellectual or, more specifically, industrial property protections. But affording programming languages the conventional arsenal of copyright law remedies is inconsistent with established legal doctrines that adjust the balance of rights between authors and publishers, on the one hand, and the public, on the other. It is also inconsistent with the public policies that those doctrines carry out. If computer programming languages are to be given any legal protection against use without consent or compensation, it should be under a new legal system suited to industrial property rather than under existing copyright law designed for beaux arts and belles lettres. Most important, any
determination establishing such a system should be made explicitly by Congress, not by courts acting under the guise of filling the interstices in existing copyright law to "clarify" supposed congressional intent.