1986

Software and Semiconductors: Why Are We Confused

John A. Kidwell

Software and Semiconductors: Why Are We Confused?

John A. Kidwell*

INTRODUCTION

During the last several years, commentators frequently have addressed the legal problems posed by the computer revolution.[1] Much of their discourse focuses on the forms of legal protection available to those who invest capital in computer technology and the special problems created by technological advances that are expensive to achieve but inexpensive to duplicate, characteristics shared by both software and semiconductor chips.[2]

The outpouring of articles on legal protection for computer technology requires some explanation. Although the subject has economic significance, this is an unsatisfactory explanation; other important issues with significant economic ramifications

* Professor of Law, University of Wisconsin-Madison. I wish to thank Elizabeth Seager, Heidi Johnston, and Jill Ramsfield, who helped with the research, and Carl Hill, Bill Whitford, Leo Raskind, Dirk Hartog, and Richard Stern, who made valuable suggestions.

1. At least one loose-leaf service and several journals are devoted to this topic. *See, e.g.*, COMPUTER L. SERV. (CALLAGHAN); COMPUTER L.J.; RUTGERS J. COMPUTERS, TECH. & L. On the question of software protection, see, e.g., Chandler, *Proprietary Protection for Computer Software*, 11 U. BALT. L. REV. 195 (1982); Davidson, *Protecting Computer Software: A Comprehensive Analysis*, 1983 ARIZ. ST. L.J. 611; Nimtz, *Development of the Law of Computer Software Protection*, 61 J. PAT. OFF. SOC'Y 3 (1979) (including an extensive bibliography); Note, *Copyright Infringement of Computer Programs: A Modification of the Substantial Similarity Test*, 68 MINN. L. REV. 1264 (1984) [hereinafter cited as MINNESOTA NOTE]; Note, *The Current State of Computer Software Protection: A Survey and Bibliography of Coypright, Trade Secret and Patent Alternatives*, 8 NOVA L.J. 107 (1983).

2. Semiconductor chips, also known as integrated circuits, are collections of transistors formed on a "chip" of silicon that work together to perform a specific electronic function. *See The Semiconductor Chip Protection Act of 1983: Hearings on S. 1201 Before the Subcomm. on Patents, Copyrights, and Trademarks of the Senate Comm. on the Judiciary*, 98th Cong., 1st Sess. 68-69 (1983) (statement of F. Thomas Dunlap, Jr., Corporate Counsel and Secretary, Intel Corp.) [hereinafter cited as *1983 Senate Hearings*].

have not attracted as much interest. Similarly, although idealistic fascination may have contributed to the spate of articles, that explanation seems insufficient as well.

This Article hypothesizes that so much is written about the problem of legal protection for software simply because the problem is peculiarly difficult, and because the nature of this difficulty has not been analyzed cogently. With respect to semiconductors, the situation has been different. Not nearly as much attention has been focused on legal protection for semiconductor chips. In addition, it appears that the legislative solution, the Semiconductor Chip Protection Act of 1984,[3] is beset with fewer conceptual difficulties than the law of software protection. This new law may substantially reduce the confusion surrounding protection for semiconductor chips. Therefore, many of the difficulties with respect to software can be illuminated by comparing the software protection problem with that posed by the semiconductor chip. The purpose of this Article is not to add to the already substantial number of recapitulations of the state of the legal doctrine, and perhaps not even to reduce the underlying difficulty, but rather to try to explain *why it is* that the questions raised by legal protection for software and semiconductor chips are especially difficult, and why, with respect to software, none of the writing to date has been able, authoritatively, to lay those questions to rest.[4]

The Article first develops a taxonomy of legal uncertainty premised on a belief that most legal uncertainties involve questions about either the *facts* of a dispute, the *values* that relevant rules embody, the *capacity* of legal institutions to implement the rules embodying given values, or, finally, the adequacy of the *language* in which applicable rules and principles are framed. After a discussion of the relationships between computers, chips, and software and a brief recapitulation of the law of software and semiconductor chips, the taxonomy is then explored in the context of software protection law, referring to

3. The Semiconductor Chip Protection Act of 1984, Pub. L. No. 98-620, tit. III, 98 Stat. 3347 (codified at 17 U.S.C. §§ 901-914 (Supp. II 1984)) (amending U.S.C. Title 17 to protect semiconductor mask works against unauthorized duplication).

4. The difficulties surrounding legal protection for computer technology cannot merely be analyzed away. One cannot always approach a legal problem, including the protection of computer technology, as if it were a verbal Rubik's cube, and by linguistic manipulations transform a jumbled patchwork of decisions into the legal equivalent of a one-colored square. Sometimes the difficulty is either outside the law, or a fundamental part of it.

the law of semiconductor chip protection for purposes of comparison. Legal questions concerning software protection are asserted to be peculiarly difficult because nearly all of the enumerated kinds of uncertanties exist and interact in the case of software, and this interaction multiplies the difficulty associated with any one. The Article concludes with the suggestion that changes in software law must be made with an understanding of the nature of the underlying difficulties, and that patience may be necessary in order to avoid multiplying the confusion.

I. PROBLEMS IN FORMULATING AND APPLYING LEGAL NORMS: A PARTIAL TAXONOMY OF UNCERTAINTY

This Article posits that the outpouring of writing concerning legal protection for software cannot be explained merely in terms of an intrinsic interest in the subject or market-generated demand[5] and is instead attributable to the particularly difficult questions software poses for the intellectual property law system. To explain and develop this premise, this Article first creates a partial taxonomy of legal difficulty. Simply put, a legal problem is regarded as difficult if the solution is uncertain. The types of uncertainty include both the uncertainty that decisionmakers face when considering a new law or the reform of existing law and the uncertainty that lawyers and judges confront in the course of adjudicating the application of what, on its face, seems a clear formulation of a legal standard.[6]

Many questions involving the creation or application of law are difficult, but an examination of cases in which the formulation or application of law is difficult or uncertain reveals that the difficulties themselves can be categorized. First, one often does not know what actually happened, in a relatively narrow sense. That is, one does not know whether or not Joe Smith was in Topeka, Kansas, on the night in question or whether or not Acme Computer borrowed the secret of slicing silicon from a competitor. Uncertainty concerning that question makes a decision about whether or not to convict Joe for armed robbery or Acme of trade secret theft difficult. This is a factual uncer-

5. *See supra* note 1 and authorities cited therein.

6. This is not an exhaustive taxonomy because it does not include situations where the difficulty is in ascertaining the applicable standard or cases involving conflicting formulations of a single standard.

tainty problem[7] and will be designated microfactual uncertainty to distinguish it from a different, but no less difficult, kind of factual problem, namely macrofactual uncertainty.

Macrofactual uncertainty is the name for ignorance about how the world works. The formulation of legal rules, and the application of those rules, is predicated on the belief that we understand causal linkages in the world. In determining whether to prohibit certain conduct, we must either believe that we *know* the consequences of that prohibition on behavior or be willing to act on our suspicions. Is increasing the penalty for drunk driving going to make people less likely to drive while drunk? Or is any such effect eliminated by an increased reluctance of police to arrest, prosecutors to charge, and juries to convict? Similarly, one may not know whether granting protection to integrated circuits encourages or discourages research and development. The list of macrofactual uncertainties is endless. Both microfactual and macrofactual uncertainties are present in software and semiconductor chip protection cases, as later sections of this Article demonstrate.

Assuming minimal confidence in the answers to the fact questions, one then confronts uncertainties about values. Assuming that we know what happened, or how the world works, what should we do? How important is it to convict the perpetrator of the robbery Joe might have committed? How certain must one be that it was Joe? Is it better, in a close case, to let the guilty go free or to convict the innocent? If one believes that theft of software raises the price of legitimate software purchases, would it still be appropriate to ignore software theft because problems of proof might lead to punishment of an unacceptably high number of nonculpable persons? Society may be unwilling to bear the social costs of effective enforcement. Sometimes it is suggested that these sorts of questions are best answered by the legislature because these questions are not properly questions of law at all, but rather questions of policy.[8] Although courts often defer to legislatures when confronted with difficult value choices, they cannot always do so

7. Some legal realists have noted that factual uncertainty has serious implications for the judicial system. *See, e.g.*, J. FRANK, COURTS ON TRIAL 14-36 (1949). "Most legal rights turn on the facts as 'proved' in a future lawsuit, and proof of those facts, in 'contested' cases, is at the mercy of such matters as mistaken witnesses, perjured witnesses, missing or dead witnesses, mistaken judges, inattentive judges, biased judges, inattentive juries, biased juries. In short, a legal right is usually a bet, a wager, on [a] chancey outcome." *Id.* at 27.

8. *See* R. DWORKIN, TAKING RIGHTS SERIOUSLY 22-28 (1977) (discussing

and certainly do not do so at the margin of interpretation and construction. Because this discussion intends to include both legislative and judicial lawmaking, the issue of where value questions are most ubiquitous can be left unanswered.

Even if one has a grasp of the facts, and knows what one's values, and hence goals, are, one may face uncertainty as to the capacity of existing institutions to accomplish those ends. If society decides that it is very important to convict the robber, is willing to expend substantial resources to obtain a conviction, and will convict if the certainty of being correct is seventy-five percent, the system may nevertheless lack the capacity to consistently implement that standard. The resources available to the system may be inadequate to generate the necessary information, or the people who judge may be incapable of reliably deciding just when it is that the certainty of proof has reached the seventy-five percent threshold. Society may decide that it wants to punish those who steal trade secrets, and pass a rule to that effect, but the administrators of the system may be incapable of deciding when trade secrets have been taken, either because of their own limited understanding of the technology or because the resources available to them are inadequate to make meaningful comparisons of the technologies of plaintiff and defendant. A particular standard may fail not because it is unclear (although it may fail for that reason) and not because it does not reflect our values (although it may fail for that reason) but rather because the institution, as it is constituted and acting within the constraints that define it, cannot do what it tries to do. Just as a watchmaker cannot repair a watch with hammer and chisel, so an overburdened court, in a case involving low stakes tried before unsophisticated jurors by mediocre lawyers making their presentations within the limits permitted by the rules of evidence, may not be capable of satisfactory adjudication. The movement from one subtle legal standard to another "improved" standard may be meaningless if the system is unable to function at the assumed level of subtlety. This inability of the judicial system to function at a prescribed level of sophistication is the problem of capability.[9]

Finally, problems may result from language itself, the indispensable tool used not only to justify a decision once made,

---

the difference between rules, principles, and policies and urging that rules and principles are law, while policies are not).

9. *See generally* R. DANZIG, THE CAPABILITY PROBLEM IN CONTRACT LAW (1978) (elaborating on the capability problem).

but also employed in the analysis itself. The creation of law inevitably involves the creation of categories and the utilization of those categories in making decisions. The categories themselves are usually specified by criteria; sometimes those criteria prove confusing because they are vague, misdescriptive, jargonistic, or technical. Uncertainty may be especially common when criteria developed in dealing with the familiar must be applied to that which is new and unfamiliar. Assuming a tariff on oranges and another on grapefruit, what is to be done with the first shipment of lemons, if a duty must be paid?[10] Similarly, how does one decide whether a computer program embodied in a chip is a copy of the program?

Another problem related to language is quite different and probably less common. As suggested above, law implies categorization and the capacity to specify the features that define the categories. If, for example, it is important to divide boards into heartwood and sapwood, it is awkward, if not impossible, to make such a distinction without a word—a name—for the feature that distinguishes between the two. It is especially awkward if the distinguishing is to be done by different people, at different locations, and at different times. In such situations, the description of the sorting process in words may be crucial. If the essential vocabulary is not available, then the likelihood of reliable differentiation is necessarily reduced. One may be unable to frame rules if one's vocabulary is unable to capture the relevant distinctions. To return to the earlier example, a society that has used wood only in fireplaces is unlikely to have words to distinguish heartwood from sapwood. Similarly, a system that has always differentiated the machine from a picture of the machine may be confounded by a picture of a circuit that is a circuit.

To recapitulate, imagine an unusual watch in need of repair. The owner takes it to a watchmaker. The watchmaker, after commenting on the novelty of the timepiece, remarks that he needs some time to determine how it works. The watchmaker has a *fact* problem. After careful examination, the watchmaker decides that he at least believes he understands

10. Arguably, this is not a problem of language but, rather a problem resulting from a legislature's failure to fully specify criteria. It is not that the word "orange" is vague. Rather, for reasons peculiar to law, the problem is that we must interpret a rule speaking only of oranges in a case involving lemons. Nevertheless, the *sentence* is ambiguous because the words orange and grapefruit imply unstated criteria, and the court must engage in the problem of interpretation implicit in language.

the workings of the watch. The watchmaker then asks what the owner means by "repair," and what degree of accuracy is required. This inquiry is necessary because, even after the watchmaker understands how the watch works, making it accurate to within one minute a month is a very different task from making it accurate to within one minute a day. The watchmaker and the owner have a *value* problem. They will need to discuss the advantages of more, as opposed to less, accuracy, and to balance the degree of accuracy against the cost of the corresponding repairs. Once the owner and the watchmaker agree on a desired degree of accuracy of, for example, one minute a month, the watchmaker may confront a *capability* problem. The watchmaker may know how the watch works, and he may understand what would need to be done in order to make it accurate to within one minute a month, but he might lack either the required tools or the manual dexterity necessary to use them.

The use of the watch repair metaphor to illustrate the *language* problem is more difficult, although the very presence of difficulty is quite revealing. Imagine, then, that the watchmaker, after having understood the problem and decided what the repair's objective should be, and having acquired the appropriate tools, explained to his apprentice what to do. "Now as you know, this is a new kind of timepiece. Most watches have a mainspring and a balance wheel. This one, though, does not have a mainspring. Instead it appears to have a magnetic gizmo that moves the hands by nudging up against this thingamabob. In a regular watch, we would adjust the tension on the mainspring. Here, I have discovered that if you just jigger with the gizmo, it appears to nudge against the thingamabob a little more slowly—rather the same effect as tightening the mainspring—and the timepiece goes a little faster. Understood?" Two days later the owner picked up the watch, repaired. On the repair slip the watchmaker had written, "Adjust tension on the mainspring—$25."

The temptation, which must be overcome if one is to proceed to the law of chips and software, is to begin to explore the nooks and crannies of this metaphor. For instance, the repair may not quite work, in the long run, because gizmos and thingamabobs are not completely like mainsprings and balance wheels. If a part in a new timepiece seems to have no analog to any part in an old watch, some watchmakers may begin to call one part a blodget, and the other a wedge, while others call the

same parts widgets and wedges. Similarly, the law may be unable to deal with a contract that specifies mainsprings but entered into by parties who really mean gizmos. The question here is whether watches are like computer programs, or judges like watchmakers.

II. THE FORMS OF LEGAL PROTECTION: A SUMMARY

There are several forms of legal protection available to those desiring to protect their investment in software or semiconductor chips. These include patent law, trade secret law, copyright law, and the new Semiconductor Chip Protection Act. Essential to an understanding of these forms of protection, however, is the following explanation of the interrelationship between computers, semiconductor chips, and computer software.

A. COMPUTERS, CHIPS, AND PROGRAMS: WHERE DOES ONE END AND THE OTHER BEGIN?

A computer is often considered to be a machine that processes information. Strictly speaking, however, the computer manipulates physical, chemical, or electrical impulses or structures that symbolize information.[11] The computer can be conceptualized as a mass of circuitry that is organized in a meaningful way but not capable of doing anything useful until programmed. This circuitry is computer hardware. Computer software is the set of instructions that controls the operation of the computer and that transform it from a powerful, but in a sense useless, combination of circuits into a device capable of performing particular tasks.[12] A semiconductor chip is a fingernail-sized bit of silicon on which an extraordinary number of transistors have been constructed and connected, so that a chip may contain a million or more electronic elements.[13] Most

11. There is a great deal of debate about whether machines may be capable of "thought." Many researchers of artificial intelligence argue that, at some level, human and artificial intelligence are the same. *See* D. HOFSTADTER, GÖDEL, ESCHER, BACH: AN ETERNAL GOLDEN BRAID 594-680 (1979); *see also* J. WEIZENBAUM, COMPUTER POWER AND HUMAN REASON 158 (1976) (quoting G.A. Miller, Language, Learning, and Models of the Mind (unpublished manuscript)) (suggesting that humans and machines can usefully be seen as "information processing systems"). For a discussion of artificial intelligence and neurological engineering, see Stevens, *Reverse Engineering the Brain*, 10 BYTE MAG. 286 (1985).

12. Gemignani, *Legal Protection for Computer Software: The View From '79*, 7 RUTGERS J. COMPUTERS, TECH. & L. 269, 278-79 (1980).

13. Boraiko, *The Chip*, 162 NAT'L GEOGRAPHIC 421, 421, 425 (1982). This

modern computers are comprised of semiconductor chips, among other elements.[14] Although for most purposes, it may be possible to treat the chip as existing on the hardware side of the equation, merely a part of the machine, one can argue justifiably that it occupies the conceptual space between the machine and the software.[15] A microprocessor chip, for example, has an architecture—a logic—that may constitute the embodiment of a particular set of instructions. It thus has the character of a programmed computer, embodying both hardware and software.[16]

The creation of software often begins with a flowchart, which is a general description of the program and is not unlike a large scale map; it gives an impression of the shape of the program, but lacks detail.[17] After the creation of the program's structure, by flowchart or otherwise, the programmer writes the program itself, usually in a high level programming language called "source code."[18] This source code is then trans-

---

article was submitted by Representative Donald Edwards during the *Copyright Protection for Semiconductor Chips: Hearings on H.R. 1028 Before the Subcomm. on Courts, Civil Liberties, and the Administration of Justice of the House Comm. on the Judiciary,* 98th Cong., 1st Sess., 355 (1983) [hereinafter cited as *1983 House Hearings*].

14. Boraiko, *supra* note 13, at 429, 438-49.

15. The distinction between hardware and software itself may prove to be illusory. *See* Sprowl, *Proprietary Rights in Programmed Computers: Looking Beyond the Hardware/Software Distinction for More Meaningful Ways of Characterizing Proprietary Interests in Digital Logic Systems,* 1983 ARIZ. ST. L.J. 785, 786.

16. One author has termed the microprocessor a "computer on a chip." Boraiko, *supra* note 13, at 421.

17. *See* Gemignani, *supra* note 12, at 272. The structure plays the role in programming that the outline plays in other kinds of writing. In fact, the process of creating software begins long before the creation of a flowchart, and people familiar with the creation of sophisticated software regard flowcharting and coding of the program to be only two of perhaps a dozen or more steps that also include evaluations of the needs of users, consideration of input and output formats, assessments of compatibility with other programs, computers, data formats, and so forth. This is important because a judicial focus on the coding aspect of software creation may be misdirected if it implies that coding absorbs the bulk of software creators' capital investment. Conversations with Carl Hill, Systems Analyst for Verex Corporation. *See also* Sprowl, *supra* note 15, at 786 (Organizations spend millions of dollars developing computer languages at levels ranging from the language of the ultimate user of the application to the language used by designers to build the circuit chips. These organizations have no legal protection for their operating languages; competitors are free to use other hardware and software to implement the same language.).

18. Gemignani, *supra* note 12, at 272. There are any number of such high level languages—COBOL, FORTRAN, Pascal, BASIC and so forth—and they

formed, often with an intermediate step, into "object code."[19] Object code is largely unintelligible to the human programmer but is needed to actually operate the computer. Object code is usually represented as a series of 1's and 0's. These 1's and 0's are analogous to the "on" and "off" settings of the computer's switches.[20]

If one conceives of a computer as an extraordinarily complicated set of electrical switches and relays, the computer program is nothing more than the list of instructions for the setting of those switches to facilitate some particular electrical manipulation. The entry of the program into the computer is nothing more than the translation of the description of the switch settings into the setting of the switches themselves. Thus, as noted earlier,[21] the instructions have been transformed into the thing itself. That is, the instructions as to switch settings have at a certain point become the switch settings. The relative ease with which this transformation is accomplished undermines the distinction between the "hardware" (the machine) and the "software" (the instructions). As a result, the distinctions become less clear and meaningful than the terms, probably chosen to imply a dichotomy, suggest.[22]

The semiconductor chip underscores the fragility of the boundaries between categories.[23] Chips are usually manufactured by a photographic process known as photolithography.[24]

---

are usually designed to facilitate the human aspect of the human-computer interaction. Tesler, *Programming Languages*, SCI. AM., Sept. 1984, at 70, 73.

19. Davidson, *supra* note 1, at 620; Sprowl, *supra* note 15, at 793-95. Sprowl notes that in a modern application program like an accounting spreadsheet, for example, there are seven layers of languages, ranging from "chip language" to the customized user language of the particular application program. *Id.*

20. Sprowl, *supra* note 15, at 788-89.

21. *See* text accompanying *supra* notes 15-16.

22. "[T]echnologically 'hardware' and 'software' are equivalent." Brief for Amicus Curiae the Association of Data Processing Service Organizations, Software Industry Association at 5, Parker v. Flook, 437 U.S. 584 (1978). Similarly, "[s]oftware can always be converted into functionally equivalent hardware, and while a minimum amount of hardware is always essential, much of what used to be pure hardware in computers is now pure software." Sprowl, *supra* note 15, at 786.

23. *See* Boraiko, *supra* note 13, at 429.

24. Photolithography is not the only technique used. H.R. REP. NO. 781, 98th Cong., 2d Sess. 20, *reprinted in* 1984 U.S. CODE CONG. & AD. NEWS 5750, 5769 [hereinafter cited as HOUSE REPORT] (All citations to the House Report are to the star print. *United States Code Congressional and Administrative News* contains the initial version of the House Report. The star print corrected typographical errors and contains three additional pages.). The process

"Masks" are used to project patterns onto a treated silicon wafer which is "developed" and becomes not the picture of the circuit, but the circuit itself.[25] It is possible to design a chip, or a part of it, which will perform a specific function, and this chip or part *is* for all practical purposes the same as a specially designed computer or a general purpose computer with a program embodied in it.[26] Returning to the metaphor of switches, some of the switches on the chip are pre-set and those settings transform that portion of the chip into the functional equivalent of the programmed computer. The semiconductor chip thus illustrates that a fine line separates the plan for the thing from the thing itself.

The diagram of the circuit transformed by the manufacturing process creates the circuit. In some cases, it might be difficult to tell whether an image was the diagram of the end product, or the end product itself. A projection of the image of the circuit can itself create the circuit. This point was made, somewhat poetically, by Frederick Turner:

> Consider the process by which a modern integrated circuit is made . . . . The technical term for this process is "photographic." An integrated circuit is essentially a very complex silicon photograph . . . . We normally think of a photographic process as one that makes pictures of things rather than things themselves. A photograph is significant only as a record; as an object it's just a bit of sticky paper. But our silicon photograph doesn't merely *represent* something; it *does* what it is a photograph of. In a sense it is a miraculous picture, like that of Our Lady of Guadalupe: it not only depicts, but does; it is not just a representation, but reality; it is not just a piece of knowledge, but a piece of being; it is not just epistemology, but ontology.[27]

The complexity of both the physical structure and the instructions should not be underestimated. A complex program might be composed of 10,000,000 lines of code, which might exceed the logical complexity of the circuitry of some computers, and be responsible for the manipulation of as many as 1,000,000,000,000 bits of data.[28] Even these numbers may underestimate the complexity of the program, as compared to the computer. Thousands of the memory locations in a computer's memory chip are virtually identical, and exist in a highly or-

---

of chip manufacturing sometimes involves the use of electron beam technology to "write" the circuit on the silicon wafer. Boraiko, *supra* note 13, at 429.

25. A more detailed description and illustration of the manufacture and capability of semiconductor chips can be found in Boraiko, *supra* note 13, at 426-27, 432-34.

26. *Id.* at 430-31; *see also supra* note 16 and accompanying text.

27. Turner, *Escape From Modernism*, HARPER'S MAG., Nov. 1984, at 47.

28. Bacon, *Software*, 215 SCI. 775, 775 (1982).

dered and systematic layout—much of which resembles a city laid out on a grid system. The lines of the program, however, tend to be relatively different from one another and are less patterned. As a result, the investment of human capital in the design of machines as complex as computers is enormous. The development of a single semiconductor chip might take three or four years, and cost $4 million, and a "family" of chips might cost as much as $80 million.[29] The investment of labor and capital in the programs may be equally substantial.

Developers of computer software and semiconductor chips are understandably eager to minimize the extent to which others can exploit that investment. Although, in some circumstances, the law of contracts or trademarks may be useful, most of the analysis of legal protection has focused on patent law, copyright law, and the law of trade secrets. Others have usefully explored the applications of those bodies of law to software[30] and semiconductor chips.[31] What follows is only a summary of doctrine necessary to provide the background for the argument that will follow. This summary is largely for the benefit of those unfamiliar with the broad outlines of intellectual property law.[32]

B. PATENT LAW: STRONG PROTECTION, BUT DIFFICULT TO OBTAIN

The most thorough protection for technological development is provided by patent law.[33] For seventeen years from the issuance of the patent, the inventor is protected against competition not only from those who copy the inventor's work, but even from those who might independently recreate the advance.[34] The scope of the patent is delineated in a detailed claim,[35] and the best mode of practicing the invention known to the inventor must be disclosed. A patent is issued only after a

29. *1983 House Hearings, supra* note 13, at 34 (statement of F. Thomas Dunlap, Jr., Corporate Counsel and Secretary, Intel Corp.).

30. *See supra* note 1.

31. *See 1983 House Hearings, supra* note 13, at 82 (statement of Dorothy Schrader, Associate Register of Copyrights); Oxman, *Intellectual Property Protection and Integrated Circuit Masks,* 20 JURIMETRICS J. 415-60 (1980).

32. It is important to note that the law has been developing quite rapidly in this area, and one must be careful of the relatively rapid obsolescence of summaries.

33. Davidson, *supra* note 1, at 230. The federal patent statutes are codified at 35 U.S.C. §§ 1-376 (1982).

34. *See* 35 U.S.C. § 271 (1982).

35. *See* 35 U.S.C. § 112 (1982).

sophisticated substantive review of the application.[36] The process is expensive, and the standards—novelty, nonobviousness, and utility—are high.[37] From the outset of the computer revolution, some have sought the protection of the patent system for software developments.[38] That early interest, however, was matched by doubts about the availability of patent protection.[39]

The patent statute provides that to be patentable a claim must be directed to a "process, machine, manufacture, or composition of matter, or . . . improvement thereof."[40] Although early Patent and Trademark Office (PTO) regulations declaring software unpatentable[41] were subsequently withdrawn,[42] the PTO has continued to interpret the rules, whenever possible, to exclude or narrow the protection of software under the patent laws.[43] In contrast, the Court of Customs and Patent Appeals (CCPA), in reviewing the PTO's decisions, has consistently taken a position more favorable to those seeking patent protection.[44] The Supreme Court seems to have moved from a position suggesting rejection, or at least skepticism, about patentability[45] to a position that supports patentability in some circumstances.[46] This story has been well documented by others in a degree of detail that would not be appropriate here.[47] Nev-

36. *See* 35 U.S.C. § 131 (1982).

37. *See* 35 U.S.C. §§ 101 (novelty and utility), 102 (no prior use or patents), 103 (nonobviousness) (1982).

38. *See, e.g., Ex parte* King, 146 U.S.P.Q. (BNA) 590 (Pat. Off. Bd. App. 1964) (rejecting a claim to a digital computer arranged to mathematically process data stored in the computer).

39. In 1966, a Presidential Commission recommended that software be unprotectable, President's Comm'n on the Patent System, *Report to Promote the Progress of the Useful Arts,* S. DOC. NO. 5, 90th Cong., 1st Sess. 20-21 (1967), and proposed legislation to that effect. *See* S. 1042, 90th Cong., 1st Sess., 113 CONG. REC. 4038 (1967), H.R. 5924, 90th Cong., 1st Sess., 113 CONG. REC. 4197 (1967).

40. 35 U.S.C. § 101 (1982).

41. 33 Fed. Reg. 15,609-10 (1968).

42. 34 Fed. Reg. 15,724 (1969).

43. For a well written history of this period, see Chandler, *supra* note 1, at 234-53; *see also* Nimtz, *supra* note 1, at 4-21 (tracing the general development of the law involving protection of computer software). For a particularly instructive recent article, see Anthony & Colwell, *Litigating the Validity and Infringement of Software Patents,* 41 WASH. & LEE L. REV. 1307, 1315-17 (1984).

44. Chandler, *supra* note 1, at 234-53.

45. *See* Gottschalk v. Benson, 409 U.S. 63, 71-73 (1972).

46. *See* Diamond v. Diehr, 450 U.S. 175, 185-88 (1981).

47. *See* Chandler, *supra* note 1, at 240-55.

ertheless, a brief recapitulation of the decisions on the issue, emphasizing those of the Supreme Court, may be useful.

Within patent law, one can encounter more than one form of claim. Hopeful claimants for patents on software or software dependent inventions have historically sought to frame their applications in ways they believed would prove acceptable to those supervising the grant of patents. One possibility in the case of computer software was to claim a particular program as a process, or part of a process, for accomplishing some end. In other words, one would claim the exclusive right to perform certain operations in a certain order. Although process claims most commonly have been used in cases involving the physical transformation of material substances, often by chemical means,[48] some of the claims to patents for software or software dependent inventions were cast in the same form. The applicant might have couched his claim in these terms. "I claim the process of performing X transformation by means of a computer which has been programmed in the following way." As discussed below, claims cast in such form faced substantial difficulty from the outset.[49]

Another route to patent protection might be for the claimant to claim a particular structure.[50] Applicants claimed patents on computers given a particular electro-mechanical "shape" by causing certain electrical "instructions" to be read into it. An applicant, in effect, claimed a patent on a complex electronic circuit having a structure consisting partly of the preconfigured computer and partly of circuits that were altered by the program. Initially, it appeared that this conceptualization was going to meet with greater success than the process claim form, at least insofar as patentability was concerned.[51]

48. J. LANDIS, MECHANICS OF PATENT CLAIM DRAFTING 71-72 (1974).

49. *See infra* text accompanying notes 52-62.

50. J. LANDIS, *supra* note 48, at 17. This is commonly the form in which a mechanical invention or a composition of matter would be claimed; for example, "I claim an alloy of bronze having as its constituent parts 10% of X metal, and 15% of Y metal, etc." *Id.* or, "I claim a mechanism for closing a container which consists of a thin circular plate, the edges of which have been bent at eighty degrees to the plane of the plate, and crimped so as to grip a protruding edge of a container." *Id.* A computer program could conceivably be described in a similar form.

51. *See In re* Johnston, 502 F.2d 765, 770 (C.C.P.A.), *rev'd on other grounds sub nom.* Dann v. Johnston, 425 U.S. 219 (1976); *In re* Prater, 415 F.2d 1393, 1403 (C.C.P.A. 1969); *see also* Dann v. Johnston, 425 U.S. 219, 225 (1976). One problem, of course, was that the choice seemed to be largely a matter of form, and judges objected to allowing major decisions to turn on the choice of a claim drafting technique. *See, e.g., In re* Johnston, 502 F.2d 765, 773

The Supreme Court first explicitly addressed the question of patentability in *Gottschalk v. Benson*.[52] In *Gottschalk*, the Court refused to decide whether programs generally were patentable, but held that the particular program being proposed was unpatentable subject matter because the patent, if issued, would preempt a mathematical algorithm and would effectively be a patent on the algorithm itself.[53] One commentator believed the decision revealed a general hostility toward patentability of programs.[54] In 1976, the Court held that a particular computer program was unpatentable because it was an "obvious" advance[55] and therefore avoided addressing the broader question of whether computer programs are patentable subject matter.[56] In the interim, the CCPA, seemingly eager to limit the reach of *Gottschalk*, decided several cases that suggested that the combination of a program with an end use that involved a physical transformation would avoid the *Gottschalk* objection.[57] In 1978, however, the Supreme Court in *Parker v. Flook*[58] reversed the CCPA's approval of an application based on just this rationale,[59] and held that the patent claim was de-

(C.C.P.A. 1976) (Rich, J., dissenting) (arguing that certain claims should not be made patentable merely because they were drafted in machine system form), *rev'd on other grounds sub nom.* Dann v. Johnston, 425 U.S. 219 (1976). It is true that a similar problem may confront an applicant wishing to patent a particular compound. One might seek to claim an invention comprising a chemical having two hydrogen atoms, six oxygen atoms, and two sodium atoms, or one might claim as an invention the process by which such a chemical could be produced. Although it might often be the case that these are two separate inventions, there are other cases in which the inventor might claim one or the other. There might be advantages associated with one approach or the other and often both claims might be made. It seems, however, that the essential patentability of a chemical advance should not turn on which claim drafting form one chooses, and this is what seems to be happening in the case of software.

52. 409 U.S. 63 (1972).

53. *Id.* at 71-72. Some of my acquaintances with a knowledge of computing expressed amazement that the claim was not rejected out of hand, given the relative triviality of the subject matter. The fact that the claim was not rejected on the merits may suggest the Court's inclination to rest its decision on procedural grounds when confronted with unfamiliar technology.

54. *See* Note, *Protection of Computer Programs: Resurrection of the Standard*, 50 NOTRE DAME LAW. 333, 339-40 (1974).

55. Dann v. Johnston, 425 U.S. 219, 228-30 (1976).

56. *See id.* at 220.

57. *See, e.g., In re* Flook, 559 F.2d 21, 22-23 (C.C.P.A. 1977), *rev'd sub nom.* Parker v. Flook, 437 U.S. 584 (1978); *In re* Deutsch, 553 F.2d 689, 692-93 (C.C.P.A. 1977); *In re* Noll, 545 F.2d 141, 148-49 (C.C.P.A. 1976); *In re* Chatfield, 545 F.2d 152, 158-59 (C.C.P.A. 1976), *cert. denied*, 434 U.S. 875 (1977).

58. 437 U.S. 584 (1978).

59. *See id.* at 590.

fective because the addition of post-solution activity to a mathematical formula is not sufficient to transform an unpatentable principle into a patentable process.[60] Proponents of program patentability were glum.

In 1981, however, the Court decided *Diamond v. Diehr*,[61] the decision that, it seemed, the proponents of patent protection for software had been awaiting. The patience of the CCPA in continuing to narrowly construe Supreme Court statements forbidding patentability of software claims had finally won the day. On facts extraordinarily similar to those in *Parker*, the Court found that a process consisting of the utilization of computer software to control a rubber molding process constituted patentable subject matter.[62]

Few infringement cases involving software dominated patents have been litigated, nor does the Patent Office possess a large body of experience in the processing of software dependent claims. The PTO has issued guidelines for applying the subject matter standard, which appear quite restrictive, and which arguably reflect a continuation of the PTO's attitude opposing such claims.[63]

The applicability of patent law to semiconductor chips is, by comparison, well settled. Unquestionably, patent protection

---

60. *See id.* at 590 (reasoning that otherwise unpatentable process or mechanism cannot be rendered patentable by the claimant's inclusion of a specified application of his intention).

61. 450 U.S. 175 (1981).

62. *See id.* at 184. In the same year the Supreme Court split 4-4 in the case of Diamond v. Bradley, 450 U.S. 381 (1981). This had the effect of affirming the CCPA's decision in a case involving a "data structure" for use in a multiprogrammed computer. *Id.* at 381.

The optimism this turnabout engendered in patent lawyers concerning the patentability of software was born out in Paine, Webber, Jackson & Curtis, Inc. v. Merrill Lynch, Pierce, Fenner & Smith, Inc., 564 F. Supp. 1358 (D. Del. 1983). Merrill Lynch's patent on the data processing methodology and apparatus for implementing its Cash Management Account survived a challenge by Paine, Webber based on an argument that the patent gave protection to a business system, and was nothing more than a patent on an accounting algorithm, while Merrill, Lynch countered by successfully arguing that the claim incorporated the "means", or apparatus, for implementing the process, and that the algorithm was not of the sort denied protection in *Gottschalk*. *Id.* at 1365-68. The District Court found that the patent claims were statutory subject matter because the claims allegedly taught "a method of operation on a computer to effectuate a business activity" while simultaneously acknowledging that the method would *not* be patentable if done by hand. *Id.* at 1369.

63. PATENT AND TRADEMARK OFFICE, MANUAL OF PATENT EXAMINING PROCEDURES § 2110 (5th ed. 1983). For an excellent account of the current situation regarding software patentability, see Anthony & Colwell, *supra* note 43, at 1317-34.

is available for some of the basic technology of the semiconductor chip. One may obtain a patent on an advance in the art of manufacturing the chip, or for a discovery of new materials from which to fabricate chips, or even for a new kind of circuit.[64] The expensive process of forming existing circuits into effective combinations, however, is assumed to be "obvious" within the meaning of patent law, and is thus unpatentable.[65] Although some still question whether chips are patentable subject matter,[66] patent law requirements of novelty and nonobviousness impose perhaps insurmountable hurdles for the great majority of chips.[67] Greater doubt exists with respect to the patentability of the masks used to create the chips.[68] In short, patent protection is regarded as inadequate to protect the capital investment of chip producers, because so much of that investment is in developments that fail to meet the substantive standards of patentability.

C. TRADE SECRET LAW: THE NEED FOR A SECRET

State trade secret law has been widely used to protect the proprietary interest of developers of both computer software and semiconductor chips.[69] Although litigation and scholarship in the patent and copyright areas has focused to date largely on the conceptual problems of whether those laws are available to

---

64. Patent law can protect the basic electronic circuitry for new microprocessors or other new such products. But patent law does not protect the particular layouts and design work performed by the different chip manufacturers in adapting those electronic circuits for a particular industrial purpose, because the creativity involved does not rise to the inventive level required by the patent laws.

HOUSE REPORT, *supra* note 24, at 3, 1984 U.S. CODE CONG. & AD. NEWS at 5752.

65. *See 1983 House Hearings, supra* note 13, at 3 (statement of Sen. Charles McC. Mathias, Jr.).

66. *See* Oxman, *supra* note 31, at 421-25 (suggesting that semiconductor chips may not fall within the statutory purview of patent law).

67. "It is almost inconceivable that the layout will be nonobvious to a person with skill in the art." *Id.* at 426; *see also 1983 House Hearings, supra* note 13, at 65-66 (statement of Gerald Mossinghoff, Assistant Secretary of Commerce and Commissioner of Patents and Trademarks) ("While a patent on the circuit could protect against the manufacture, use or sale of the circuit, the circuits in chips are usually well-known and unpatentable.").

68. *See* Oxman, *supra* note 31, at 421-25 (citing 35 U.S.C. § 101 (1982) in suggesting that masks fail to satisfy the statutory requirement of being a "new or useful process, machine, manufacture, or composition of matter, or any new or useful improvement thereof").

69. R. MILGRIM, TRADE SECRETS app. A-1, at 2-3 (1982); Davidson, *supra* note 1, at 717-19.

proprietors, these issues have not arisen in the trade secret context. It has always been clear that the law of trade secrets was appropriate to preserve a proprietor's right to exploit software or circuit designs.[70] This is not to say that serious and difficult issues do not arise in the context of the utilization of trade secret protection, but rather that those issues have not involved the threshold question of whether software or chip designs are protectable subject matter.[71]

Owners of trade secrets are entitled to protection against those who discover their secrets by improper means, which might include acts as bold as burglary, but usually involve violation by an employee of an express or implied obligation to preserve the secrets of the employer. The most difficult issues in trade secrets litigation involve such questions as whether the information has in fact been kept sufficiently secure within the company or industry to qualify as a secret,[72] or whether a former employee has actually used a trade secret as opposed to utilizing the skills and training which were acquired in the course of the employment and which the employee is entitled to carry to a subsequent job.[73]

---

70. R. MILGRIM, *supra* note 69, app. A-1, at 2 (1982); Davidson, *supra* note 1, at 701-13.

71. The most widely used definition of "trade secret" is found in the RESTATEMENT OF TORTS § 757 comment b (1934). It states:

> A trade secret may consist of any formula, pattern, device or compilation of information which is used in one's business, and which gives him an opportunity to obtain an advantage over competitors who do not know or use it. It may be a formula for a chemical compound, a process for manufacturing, treating or preserving materials, a pattern for a machine or other device, or a list of customers.

Id. Software would seem to fit comfortably within the definition.

72. *See* Jostens, Inc. v. National Computer Sys., 318 N.W.2d 691, 700 (Minn. 1982) (holding that plaintiff failed to show that it had intended to keep its computer systems used for the design and manufacture of class ring molds, a trade secret); Computer Print Sys. v. Lewis, 422 A.2d 148, 153-54 (Pa. Super. Ct. 1980) (holding that specific computer programs designed to expedite direct mail advertising had remained protectable trade secrets).

73. *See* Structural Dynamics Research Corp. v. Engineering Mechanics Research Corp., 401 F. Supp. 1102, 1116 (E.D. Mich. 1975) (holding that there was a violation of a trade secret when two former employees of plaintiff corporation who had signed a "Confidential Information Agreement" later joined defendant corporation and gave it information concerning an isoparametric element computer system developed by plaintiff corporation); *cf.* Sperry Rand Corp. v. Rothlein, 241 F. Supp. 549, 564 (D. Conn. 1964) (holding that though defendants were entitled to take with them the skills acquired as employees of plaintiff company, knowledge of the final results of the end product of their work, including the manufacture of a silicon alloy junction transistor, constituted a trade secret which was violated when defendants conveyed it to one of plaintiff's competitors).

Although trade secret protection may prove useful to any number of developers of software or semiconductors, it offers neither protection against independent innovation nor protection against duplication of a rightfully acquired copy. The formula for Coca-Cola is proprietary information only as long as no one can independently recreate the flavor, and only as long as Coca-Cola can maintain the secrecy of its formula. Thus, although trade secret law may protect software used only by the developer or licensed to a limited number of persons and designers of integrated circuits during the time that the chip is in the design and development stages, trade secret law offers little comfort to those who market software more widely or to chip manufacturers once the chip is in production and distribution. Even though trade secret protection might prove quite acceptable to developers of software who retain complete control over their product, it is less helpful to sellers of software used for computer games, or in connection with mini or microcomputers. Few chip manufacturers would ever expect to keep their chip designs within the mantle of trade secret protection, because few chips are customized for internal use by either the developers themselves or their customers.[74]

D. COPYRIGHT LAW: A HOST OF PROBLEMS

Copyright law provides protection to the "writings" of authors. The term "writings" has been expansively defined to include not only books, but paintings, motion pictures, sculpture, maps, speeches, and computer programs.[75] To qualify for protection a work must possess a minimal degree of originality and be fixed in a tangible form.[76] The degree of originality required is modest; telephone books and art reproductions are sufficiently original to qualify for copyright protection.[77]

The formalities associated with obtaining copyright protection are minimal. Copyright arises upon "fixation" of the work.[78] If the work is published, a simple notice is affixed.[79]

74. This is not to say that chips are not customized. They are, but such customization is normally for ultimate mass distribution as control devices for items such as consumer products. Oxman, *supra* note 31, at 418.

75. 17 U.S.C. §§ 102, 117 (1982).

76. 17 U.S.C. § 102 (1982).

77. *See, e.g.*, Hutchinson Telephone Co. v. Fronteer Directory Co., 770 F.2d 128, 132 (8th Cir. 1985) (telephone directory held copyrightable); Alva Studios v. Winninger, 177 F. Supp. 265, 267 (S.D.N.Y. 1959) (art reproductions copyrightable when skill and originality were required to produce the replica).

78. 17 U.S.C. § 102 (1982).

Registration of the copyright must precede enforcement.[80] Copyright owners are granted, by the statute, certain exclusive rights including the right to reproduce, distribute, perform, and display the work, as well as the right to prepare derivative works.[81] These rights are subject to enumerated limitations that are designed to promote public objectives.[82] One who uses a copyrighted work is not guilty of infringement unless that use violates one of the exclusive rights; the exclusive rights are not exhaustive of all uses.[83] Like trade secret law, and unlike patent law, copyright provides no protection against independent creation of a similar, or even identical, work.[84]

When copyright law encounters computer technology, difficult issues arise. Questions such as whether a computer program is a "writing" of an author or whether, even if copyrightable, the program is subject to protection in its object code form, or what degree of similarity is required to prove infringement, present complex queries for analysis under the copyright laws. The current copyright law, enacted in 1976 and amended in 1980,[85] makes it clear that copyright protection may extend to computer programs.[86] Just what that *means*, however, is not so clear.

Copyright protection is denied to ideas, processes, and the like[87] and extends only as far as the expression of the idea or

79. 17 U.S.C. § 401 (1982).

80. 17 U.S.C. § 411 (1982).

81. 17 U.S.C. § 106 (1982).

82. 17 U.S.C. §§ 108-118 (1982).

83. 2 M. NIMMER, NIMMER ON COPYRIGHT § 8.01 (1985).

84. Even if the defendant is guilty of a prima facie violation of an enumerated exclusive right, the defendant may claim the benefit of the "fair use" privilege of § 107, 17 U.S.C. § 107 (1982). Section 107 is clearly intended to permit at least de minimus violations. Although the meaning of the "fair use" provision is still being grappled with, *see* 3 M. NIMMER, *supra* note 83, at § 13.05, it is likely that the fair use privilege will eventually be construed to permit, in addition to de minimus borrowings, those uses that are in the public interest but that will not be legitimatized by license from the copyright proprietor either because the author wishes to be free from criticism or parody, or because the transaction costs associated with seeking permission are so high that the user would prefer to forego use rather than undergo the costs of obtaining a license. *See* Sony Corp. of Am. v. Universal City Studios, 457 U.S. 417, 433 (1984) ("All reproductions of the work, however, are not in the exclusive domain of the copyright owner; some are in the public domain.").

85. Act of Oct. 19, 1976, Pub. L. No. 94-553, 90 Stat. 2541 (1976), *amended by* Act of Dec. 12, 1980, Pub. L. No. 96-517, 94 Stat. 3015 (1980) (codified as amended at 17 U.S.C. § 101-810 (1982)).

86. 17 U.S.C. §§ 101, 117 (1982).

87. 17 U.S.C. § 102 (1982).

the description of the process.[88] This has been construed to mean that the idea must remain available for public use.[89] Therefore, if only a very limited number of possible expressions of an idea are possible, copyright cannot be extended to those expressions.[90] Without this requirement, it would be possible to substantially restrict access to the idea by claiming a copyright in one or more of the few possible variations in expression of the idea.[91] Furthermore copyright is not available for utilitarian creations like car bodies, shoes, or lamps—or more precisely, it is available only for the separable, decorative features of utilitarian creations like the hood ornament, or bulldog image on the belt buckle, or statuette serving as a lamp base.[92] As a result, it could be argued that programs, at least as embodied in read only memory (ROM)[93] or random access memory (RAM)[94] chips, are not copyrightable because their "literary" aspects are inseparable from their functional, utilitarian features.[95] Third, there has been much discussion of whether programs in object code were "copies" within the meaning of the statute.[96] Finally, the degree of similarity necessary to constitute infringement is likely to prove problematic in cases involving software.[97] Even literal copying could arguably be permitted in some situations.[98]

---

88. *See* 2 M. NIMMER, *supra* note 83, § 2.03[D].

89. *See* 3 M. NIMMER, *supra* note 83, § 13.03[A].

90. *See* 3 M. NIMMER, *supra* note 83, § 13.03[A].

91. *See* Morrissey v. Proctor & Gamble Co., 379 F.2d 675, 678-79 (1st Cir. 1967).

92. *See* 17 U.S.C. §§ 101, 113 (1982).

93. A read only memory (ROM) chip is one from which the computer can only read stored information; it cannot change or "write" new information onto the chip.

94. A random access memory (RAM) chip is capable of receiving and temporarily storing data, and allows changes to be made on what is stored on the chip.

95. *See* Davidson, *supra* note 1, at 671-72; Samuelson, *CONTU Revisited: The Case Against Copyright Protection for Computer Programs in Machine Readable Form*, 1984 DUKE L.J. 663, 739-41.

96. *See* Data Cash Sys. v. JS&A Group, Inc., 480 F. Supp. 1063, 1067-69 (N.D. Ill. 1979), *aff'd on other grounds*, 628 F.2d 1038 (7th Cir. 1980). Apple Computer, Inc. v. Franklin Computer Corp., 714 F.2d 1240 (3d Cir. 1983), *cert. denied*, 464 U.S. 1033 (1984), may have laid this argument to rest, if only because it seems a "practical" result. *See id.* at 1248-49. The court held that a computer program in object code embedded in a semiconductor chip is an appropriate subject for copyright. *Id.* at 1249.

97. *See* SAS Inst., Inc. v. S&H Computer Sys., 605 F. Supp. 816, 829 (M.D. Tenn. 1985) (finding infringement in case of nonliteral similarity); Chandler, *supra* note 1, at 228; Davidson, *supra* note 1, at 693.

98. Morrissey v. Proctor & Gamble Co., 379 F.2d 675, 678-79 (1st Cir. 1967)

These difficulties led to the conclusion that copyright was not available to protect circuit designs embodied in semiconductor chips. The Copyright Office had refused to register chips as copyrightable works,[99] on the ground that they were primarily utilitarian objects with no separable nonutilitarian features.[100] Litigation based on copyright claims in chip designs embodied in drawings proved inconclusive.[101] Concerns about protection for chips, of course, led to the enactment of the Semiconductor Chip Protection Act.

E. SEMICONDUCTOR CHIP PROTECTION ACT OF 1984

The most recent addition to the forms of legal protection for intellectual property is the Semiconductor Chip Protection Act of 1984.[102] Congress passed the Act in response to a need for chip protection and findings that existing forms of protection were inadequate.[103] The Act grew out of efforts to amend the copyright statute to extend copyright protection to semiconductor chips, but the new form of protection is sufficiently different from copyright protection to be designated a *sui generis* form of protection.[104]

The Act extends protection to the original[105] configuration

---

(finding that contest rules were so straightforward and simple that they were an uncopyrightable form of expression).

99. In 1977, Intel Corp. brought a suit to compel the registration of a design of a chip. The case was dismissed without prejudice and without disposing of the option of registrability. Intel Corp. v. Ringer, No. C77-2848-RHS (N.D. Cal. 1978); *see* Wilson & LaBarre, *The Semiconductor Chip Protection Act of 1984: A Preliminary Analysis*, 67 J. PAT. OFF. SOC'Y 57, 63 (1985) (discussing *Intel Corp. v. Ringer*).

100. With some misgivings, the Copyright Office began accepting mylar sheets and photolithographic masks for semiconductor chips as technical drawings. *See Copyright Protection for Imprinted Design Patterns on Semiconductor Chips: Hearings Before the Subcomm. on Courts, Civil Liberties, and the Administration of Justice of the House Comm. on the Judiciary*, 96th Cong., 1st Sess. 6 (1979) (statement of Jon Baumgarten, General Counsel, U.S. Copyright Office) [hereinafter cited as *1979 House Hearings*].

101. *See* Wilson and LaBarre, *supra* note 99, at 64 n.29 (citing Zilog v. Nippon Elec. Corp., C-83-1241-WHO (N.D. Cal. 1983); Intersil, Inc. v. Teledyne Corp., No. C-82-4187-WHO (N.D. Cal. 1982); U.S. Int'l Trade Comm'n, Inv. No. 337-TA-153 (1983)). All the cases were settled prior to decision. *See* Wilson & LaBarre, *supra* note 99, at 64.

102. For a summary of the background of the Semiconductor Chip Protection Act of 1984, 17 U.S.C. §§ 901-914 (Supp. II 1984), see Wilson & LaBarre, *supra* note 99, at 59-69.

103. HOUSE REPORT, *supra* note 24, at 3-4, 1984 U.S. CODE CONG. & AD. NEWS at 5752-53.

104. *See id.* at 5, 1984 U.S. CODE CONG. & AD. NEWS at 5754.

105. The originality requirement under the Act appears more demanding

of the chips themselves, called "mask works" in the Act.[106] The chips themselves are also protected, rather than just the "masks" conventionally used to create them using photolithographic techniques, because chips can currently be created without the use of a mask by electron beam etching; other techniques may be used in the future.[107] Mask work owners have exclusive rights to reproduce the mask work, to import or distribute semiconductor products embodying the mask work, and to induce others to reproduce, import or distribute the chip products.[108] The protection endures from ten years from the date of registration or exploitation, whichever comes first.[109] The rights granted to the chip's creator are subject to a privilege in others to use reverse engineering to replicate the circuitry for purposes of study or incorporation in another original mask work,[110] as well as to a compulsory licensing scheme to protect innocent infringers.[111]

III. THE TAXONOMY OF UNCERTAINTY APPLIED

As has already been suggested, the law of intellectual property as applied to software, as well as to integrated circuits, is confusing because both subject matters suffer from nearly all of the kinds of uncertainties previously identified.[112] This portion of the Article applies the taxonomy of uncertainty to issues of software and integrated circuit protection. It concludes that many distinctions in the law of intellectual property dissolve under analysis when applied to software and chips.

A. FACT PROBLEMS: UNFAMILIARITY WITH SOFTWARE AND CHIP TECHNOLOGY AND PROOF OF VIOLATIONS OF THE SEMICONDUCTOR CHIP PROTECTION ACT

Questions of protection for software and semiconductor chips raise problems of the kind earlier identified as microfactual.[113] First, as to software, in any given case, the complexity

than that under copyright law, since it does not protect designs that are staple, commonplace, or familiar. *See* 17 U.S.C. § 902(b) (Supp. II 1984); Wilson & LaBarre, *supra* note 99, at 79-80 nn. 111-15.

106. 17 U.S.C. §§ 901(a)(2), 902 (Supp. II 1984).
107. Wilson & LaBarre, *supra* note 99, at 70 n.66.
108. 17 U.S.C. § 905 (Supp. II 1984).
109. 17 U.S.C. § 904 (Supp. II 1984).
110. 17 U.S.C. § 906 (Supp. II 1984).
111. 17 U.S.C. § 907 (Supp. II 1984).
112. *See supra* text accompanying notes 5-10.
113. *See supra* text accompanying note 7.

of and unfamiliarity with either the idea of software itself, or with some particular software, may increase the likelihood of judicial confusion; this alone may be sufficient to make the case difficult. It is often noted that lawyers and judges do not presently possess a sound understanding of computers.[114] In fact, as suggested later, this complexity and unfamiliarity may be so pervasive that the factual difficulties in a given case become emblematic of a general capability limitation of the judicial system.[115]

Some of the factual uncertainties that arise in disputes about software are no different than the factual questions that are nearly always present in litigation. Witnesses lie or become confused, documents are lost or destroyed, memories fade. Other factual difficulties, however, are a function of the complexity and unfamiliarity of this particular subject matter. Computer chips and software systems are hard to understand; this complexity itself may make fact-finding more difficult. In cases involving large systems, which will have been the work product of many people working over time, establishing the genealogy of a program or circuit may be very difficult. In addition, comparisons of one program to another may prove taxing to the memory, attention span, and cognitive capacity of a judge or a jury. Both kinds of factual problems—genealogy and

114. Sometimes this point is made diplomatically: "Today, most legislators, judges, and government officials have little personal knowledge of computer systems and software and, therefore, have at best an inadequate, and at worst a wrong understanding." Bosworth, *Hardware and Software—What Are They?*, in UNIVERSITY OF SOUTHERN CALIFORNIA LAW CENTER, FOURTH ANNUAL COMPUTER LAW INSTITUTE 1, 26 (1983). Others are less kind: "Now it could be that the ineffable pleasure of working with nincompoops whose technological grasp of their cases has ranged from inaccurate to nil has been mine alone." Lecht, *DP Lawyers: Hessians of U.S. Tech Revolution*, COMPUTERWORLD, May 16, 1983, at 784. Professor Pamela Samuelson's article is one of the best expositions of the thesis that the inability of lawyers and judges to understand software has had substantial effects on the development of software law. *See* Samuelson, *Creating a New Kind of Intellectual Property: Applying the Lessons of the Chip Law to Computer Programs*, 70 MINN. L. REV. 471, 504-06 (1985).

115. *See infra* text accompanying notes 156-184. There is certainly a relationship between fact problems and capability problems. The relationship, however, is not an identity. A particular case may be complex and present difficulties because of quarrels about the facts. It may not be the case, however, that the pattern of factual difficulties is so general as to reflect on the capability of the system to deal effectively with such cases. With respect to computer software cases, after the judiciary has eliminated the pervasive unfamiliarity that may presently constitute a capability problem, individual disputes will continue to pose factual difficulties.

similarity—are present in patent, copyright, and trade secret cases, because all require some degree of similarity[116] and some determination of the genealogy of the program. Both copyright and trade secret cases involve questions of whether access existed and copying occurred.[117] Even though trade secret protection might pose fewer conceptual problems than the other forms of protection, it is true in the software trade secret case, as in nearly any other trade secret case, that there may be substantial uncertainties as to whether any borrowing occurred, or whether the borrowing led to illegitimate exploitation. No one could testify that they saw the defendant walk out of a factory with the idea in his head. Although some borrowings may involve physical misappropriations, as many may consist of thefts using the vehicle of the mind, the transmission of software over phone lines, or physical removal on ubiquitous magnetic media that may subsequently be erased.

Troublesome questions of fact also have arisen, and will continue to arise, in cases involving integrated circuits, but these are arguably less difficult than those arising in software cases. The Semiconductor Chip Protection Act seems to forbid only copying—consequently the problems of genealogy, access, and similarity will be present, just as under conventional copyright and trade secret law.[118] One of the questions that arose in the debate about the Act was how one could establish copying, and how it could be distinguished from legitimate "reverse engineering."[119]

There are several possible methods available to establish copying. One might sometimes rely on an imperfection in one

116. *See supra* notes 33-101 and accompanying text. The particular standards of similarity differ, of course.

117. The development of a critical system for an IBM compatible computer by Phoenix Software Systems, Inc. is a fascinating tale of the efforts of a company to avoid the uncertainties of copying and access. A designer, who was unfamiliar with the IBM system, designed a functionally equivalent system that was not a duplicate of the IBM system. He was denied access to information about the IBM system deliberately and was instead given only certain critical constraints within which to work. The system that resulted is "guaranteed" not to infringe any IBM coyrights. *PC Compatibility: Out of the Closet*, MICROCOMPUTING MAG., Aug. 1984, at 66.

118. Interestingly, the language of the Act does not mention copying. Instead, it makes reference to reproduction. *See* 17 U.S.C. §§ 901(a), 905, 906 (Supp. II 1984). Richard Stern's article in this issue, Stern, *Determining Liability for Infringment of Mask Work Rights Under the Semiconductor Chip Protection Act*, 70 MINN. L. REV. 271, 307-13 (1985) addresses this question.

119. *1983 House Hearings, supra* note 13, at 36-37.

chip that appeared in a competitor's chip.[120] This, however, would establish copying but not necessarily infringement.[121] In other cases, one might draw inferences from a degree of similarity that could not plausibly be coincidental. In most of the cases, however, proving copying might actually be accomplished with relative ease because literal identity would ordinarily be expected in a comparison of the original and copied chips. Furthermore, the cost advantages of "piracy" are most substantial in the case of the replication of the entire pattern. To make a partial copy, it would be necessary to deconstruct the first chip in order to determine the function of the copied, or omitted, portion; this deconstruction would be much more expensive than duplication of the entire pattern, even if it were not a form of permissible reverse engineering.[122] Thus, the most common form of illegitimate copying, duplication of the entire pattern, would not likely prove very difficult to establish.

The House subcommittee received testimony concerning the ability to distinguish between unlawful copying and lawful reverse engineering.[123] In cases involving identical patterns, some inquiry into the process which led to the similarity would be required to establish either improper copying or legitimate reverse engineering.[124] It was not clear whether this was a purely hypothetical concern, or whether it would on some occasions be commercially practicable for a competitor to undertake the expense of reverse engineering the chip and then creating a

120. *See 1983 House Hearings, supra* note 13, at 36-37 (testimony of F. Thomas Dunlap, Jr., Corporate Counsel and Secretary, Intel Corp.) (describing such a situation).

121. The competitor might argue that the error was necessary in order to duplicate accurately the functional characteristics of the chip. If the error was deliberately reproduced after the first chip had been subjected to "reverse engineering," there would be no violation of the Act. Officials from Nippon Electric Corp. (NEC) offered just such a justification for copying the error on the Intel chip: "If you're not 100 percent identical, you're dead. If you take the fatal flaw out, it wouldn't be compatible." Morgan, *Battling to Innovate and Emulate:* Intel v. Nippon Electric, Wash. Post, May 2, 1983, at A10, col. 1. Whether or not that justification makes sense in the NEC case, it may in other situations. *See generally* the discussion in R. STERN, SEMICONDUCTOR CHIP PROTECTION § 1.2(B) (1986).

122. *See, e.g., 1983 House Hearings, supra* note 13, at 28 (testimony of F. Thomas Dunlap, Jr., Corporate Counsel and Secretary, Intel Corp.) (pointing out that the cost of reverse engineering a chip that cost $4,000,000 to develop would be approximately $1,000,000 while the cost of a photographic reproduction would be only $100,000).

123. *See 1983 House Hearings, supra* note 13, at 35-38 (testimony of F. Thomas Dunlap, Jr., Corporate Counsel and Secretary, Intel Corp.).

124. *Id.*

substantially similar chip. F. Thomas Dunlap, Jr., Corporate Counsel and Secretary of Intel Corporation, suggested that in all cases of legitimate reverse engineering there would be a "paper trail"—a great deal of documentation which could be relied upon to establish that lawful means had been used to produce what was assumed to be a substantially similar circuit.[125] Moments later, Dunlap stated, "the short answer in terms of the majority of the cases of copying, it is going to be something like obscenity. You will know when you see it."[126] To this Representative Barney Frank responded, "That would make me very nervous . . . . I like the long answer better."[127] Dunlap thereafter decided he liked the longer answer better as well.[128] Evidence of a "paper trail" thus became the answer to the question of how legitimate reverse engineering is distinguished from unlawful piracy.[129] Just how this will work in practice remains uncertain.

What is the significance of the likelihood that litigation concerning software and integrated circuits may involve a degree, and perhaps a relatively high degree, of microfactual uncertainty? At the outset, it may be appropriate for lawyers preparing cases to be sensitive to the special demands of complex cases, both in terms of preparing evidence and preparing their clients to deal with the realization that the court is having difficulty understanding the factual presentations. In addition, courts and legislatures should take the likelihood of factual complexity into account in their articulation of standards so as to avoid creating capability problems. It is important to avoid creating standards that require, for their meaningful application, an order of precision of fact finding that exceeds the capability of the system.

Macrofactual uncertainties may be of even greater interest and importance when considering the adequacy of present laws and the prospect of shaping them for the future—whether that shaping is to be accomplished by judicial decision or legislative enactment. For example, proponents of vigorous protection for software have asserted that such protection is critical if an appropriate amount of capital is to be invested in software crea-

125. *Id.* at 36.
126. *Id.* at 37.
127. *Id.* at 37.
128. *Id.*
129. HOUSE REPORT, *supra* note 24, at 21, 1984 U.S. CODE CONG. & AD. NEWS at 5770.

tion and dissemination.[130] They argued that a high degree of misappropriation was having, and would have, a negative effect on software innovation.[131] This is certainly the argument made to justify the Semiconductor Chip Protection Act.[132] The validity of this argument, however, is uncertain. Although borrowing that yields increased competition may change the profitability of a particular investment and the willingness to continue such investments in the future, it is equally certain that borrowing in some situations stimulates innovation and reduces the associated costs.

Imagine, for example, software developers A and B who comprise a well-defined fragment of the industry. Which case will produce greater innovation? One in which A must tolerate a substantial amount of borrowing by B, but A is free to borrow reciprocally from B and will not face the uncertainty that would accompany attempts to comply with strict anti-borrowing rules? Or, in contrast, a case in which A and B could each be assured of the exclusive opportunity to exploit their respective developments but could not engage in mutual borrowing? It is important to remember that innovators and borrowers are not mutually exclusive classes. Most innovators borrow; if borrowing were not permissible, to at least some degree, it would substantially retard the rate of innovation and increase its cost. It is, of course, true that some who borrow do not innovate—the pirates![133] Even pirates, however, are a disparate lot. At one extreme, commercial pirates engage in massive copying and no innovation in the hope of realizing large sums of money before being stopped. At the other extreme, hobbyists make copies only to back up legitimately purchased copies. In between are people who copy programs for the challenge, people

130. *See, e.g.*, Brief for Amicus Curiae the Association of Data Processing Service Organizations, Software Industry Organization at 8-10, Dann v. Johnston, 425 U.S. 219 (1976) (asserting that pioneering, independent software producers would gain access to capital markets only where protection for products is available); Brief of Amicus Curiae Whitlow Computer Systems, Inc. at 9-12, Gottschalk v. Benson, 409 U.S. 63 (1972) (lack of patent protection would jeopardize continued orderly growth of the computer programming industry).

131. *See, e.g.*, Brief for Amicus Curiae Whitlow Computer Systems, Inc., at 14-15, Gottschalk v. Benson, 409 U.S. 63 (1972) (patent protection necessary to insure crucial incentives for industry development).

132. *See, e.g., 1983 House Hearings, supra* note 13, at 3 (statement of Sen. Charles McC. Mathias, Jr.); *id.* at 177-78 (Letter from Warren Davis, Director, Government Relations, Semiconductor Industry Association).

133. *See Software Rentals: Piracy is the Hot New Issue*, BUS. WK., Aug. 1, 1983, at 90-91 (copying of programs by people who rent or borrow programs seen as a form of piracy).

who copy programs because they do not have the money to buy them, and user groups that assist one another in making copies they could buy, but would prefer to take for free. Computer magazines are filled with letters from software authors and copyists, rationalizing their respective positions and often engaging in name-calling, finger-pointing, and self-justification in the process. These colloquies usually fail to make any meaningful distinctions between *pirates* and pirates.

The Semiconductor Chip Protection Act provides a useful example of the factual debate over the relationship between protection and innovation. Representative Robert Kastenmeier, and others, more than once asked witnesses how it was possible to explain the pressing need for protective legislation when the semiconductor industry had grown so dramatically without it in the last fifteen years. The replies suggested, without much elaboration, that circumstances had changed.[134] This response contrasts with earlier differences of opinion within the semiconductor industry that led to the failure of a previous attempt to create protection for chip innovators.[135] Some in the industry believed that protection would defeat healthy and necessary borrowing, while others claimed that it would not.[136] The volatility of either the facts or the chip-producers' perceptions of the facts is, of course, a matter of concern when one is attempting to draft legislation that cannot easily be changed to adjust to changing circumstances.

It is clear that a great deal of the evidence about the amount of copying taking place, the reasons for that copying, and its economic effects, is impressionistic and speculative.[137]

---

134. *See 1983 House Hearings, supra* note 13, at 44 (testimony of F. Thomas Dunlap, Jr., Corporate Counsel and Secretary, Intel Corp.).

135. *See id.* at 43; *see also supra* note 130.

136. *See, e.g., 1979 House Hearings, supra* note 100, at 52 (statement of National Semiconductor Corp.) (ease of copying important to health and vitality of industry); *id.* at 74 (statement of General Instrument Corp.) (patent protection would give innovator far more protection than necessary and would stultify the rapid diffusion of innovative technology necessary to health of the industry).

137. *See, e.g.,* Note, *Semiconductor Chip Protection: Changing Roles for Copyright and Competition,* 71 VA. L. REV. 249, 258 (1985) [hereinafter cited as VIRGINIA NOTE] (noting that copying and reverse engineering—using the terms interchangeably—are "almost universal in the industry because technological change is so rapid."); Note, *The Policy Implications of Granting Patent Protection to Computer Software: An Economic Analysis,* 37 VAND. L. REV. 147, 180 (1984) [hereinafter cited as VANDERBILT NOTE] (providing a model of the tradeoffs and concluding that protection is appropriate, but offering little in the way of empirical basis for the conclusions). *But see 1983 House Hear-*

Well-established commercial innovators can be expected to urge tough anti-copying laws, with an eye toward maximizing the extent to which they can exploit their creations. Conversely, borrowers can be expected to argue that the prior innovators are excessively greedy. All the participants in the debate seem to rely on different versions of the facts; it is no wonder that policy makers may be perplexed.

Even if one has the facts about software borrowing and a thorough understanding of the factual relationships between various possible standards and the costs and rates of innovation, one would still be faced with value questions. In order to frame appropriate rules one needs to understand not just what the rules will do, but what one wants them to do. Even if one knows how to get to Oshkosh, the question remains, do you want to go there?

B. THE VALUE PROBLEM: WHAT ARE OUR OBJECTIVES AND WHO CAN BEST IMPLEMENT THEM?

Making the rather strong assumption that one knows how the world works when it comes to computer software and semiconductor chips, the formulation of standards must await an identification of objectives. Choices must be made between allowing relatively more freedom of borrowing, substantial utilization of secrecy as a means of protection, and relatively more price competition, or permitting somewhat less borrowing, less secrecy, marginally higher investment in development, and higher prices. Nearly every software protection or chip protection decision will have some marginal effects of this kind. To some the choice is not difficult, but an in-depth examination of the record of opinion reveals the expected uncertainty.[138]

Some articles have identified the trade-offs between rules giving more or less protection to software developers.[139] The

ings, supra note 13, at 179 (statement submitted by the Semiconductor Industry Association entitled "The Economic Effects of Chip Piracy on the U.S. Semiconductor Industry") (offering an insightful, concrete, and factually-based analysis of the extent of chip piracy and its deleterious effects on innovation in the semiconductor industry).

138. *See infra* note 139 and authorities cited therein. The costs and benefits of the various forms of protection—patent, copyright, trade secret law, and the Semiconductor Chip Protection Act—need to be evaluated separately because there is at least some difference of opinion as to the need for more extensive protection.

139. *See, e.g.*, Gemignani, *supra* note 12, at 311 (patent protection would cause few additional administrative problems for courts and Copyright Office, but would stifle competition); Keefe & Mahn, *Protecting Software: Is It*

National Commission on New Technological Uses of Copyrighted Works (CONTU) addressed the economic effects of assuring copyright protection for computer software in its report[140] and concluded copyright protection was desirable.[141] As previously noted,[142] there was once a split within the semiconductor industry as to the value of more thorough protection for semiconductor chips; the industry apparently united only when market conditions changed.[143] In addition, some have argued that the burdens of the chip law may exceed its benefits.[144]

Finally, briefs filed in Supreme Court cases addressing the software issue in the context of patentability have reflected the diversity of opinion within the computer industry.[145] For example, some on the hardware side of the industry were, at one time, concerned that more rigorous protection for software might not be in their interests, and they argued that development might be hindered if the free flow of information were

---

Worth All the Trouble?, 62 A.B.A. J. 906, 907 (1976) (trade secret laws might afford sufficient protection for the industry without unnecessarily restricting ideas, but such laws can be costly and cumbersome); Nycum, *Legal Protection for Computer Programs*, 1 COMPUTER L.J. 1, 55-57 (1978) (program patents might aid the establishment of orderly software markets, but could also stifle program development); Scaffetta, *Computer Software Protection: the Copyright Revision Bills and Alternatives*, 8 J. MAR. J. PRACT. & PROC. 381, 393-95 (1975) (patentability will present practical impediments to all users of computers, while lack of patent protection will diminish the staying power of minority groups in the industry); VANDERBILT NOTE, *supra* note 137, at 175-76 (patent protection would cause increase in technological knowledge in software industry but could give patent holders power to charge monopoly prices).

140. NATIONAL COMM'N ON NEW TECHNOLOGICAL USES OF COPYRIGHTED WORKS, FINAL REPORT 23-27 (1979). Congress established the National Commission on New Technological Uses of Copyrighted Works (CONTU) in 1975 to study the problems of photocopying and computer programs as they relate to copyright protection. *Id.* at 5-6.

141. *Id.* at 9-14. *But see id.* at 27-30 (dissenting opinion of Comm'r John Hersey) (arguing that existing laws accorded satisfactory protection for computer programs).

142. *See supra* note 135.

143. *1983 House Hearings, supra* note 13, at 44 (testimony of F. Thomas Dunlap, Jr., Corporate Counsel and Secretary, Intel Corp.).

144. *See, e.g.*, VIRGINA NOTE, *supra* note 137, at 292-95.

145. *Compare* Brief for Amicus Curiae the Computer Business Equipment Manfuacturer's Association at 20-22, Dann v. Johnston, 425 U.S. 219 (1976) (computer industry has developed in an atmosphere of free exchange of programs and nature of programs makes them ineligible for patentability) *with* Brief for Amicus Curiae the Association of Data Processing Service Organizations and Software Industry Associations at 8-9, Dann v. Johnston, 425 U.S. 219 (1976) (patents on software create incentives for innovation and investment).

impaired.[146] Others, at the same time, quarreled with the rosy picture of the growth of the computer industry painted by the dominant manufacturers.[147]

A related and substantial question involving conflicting values concerns which of the legal institutions with the potential power to make decisions that are pro or anti-borrowing should exercise that power. The preservation of the balance of decision-making authority mandated in the Constitution suggests that courts should be reluctant to make decisions involving fundamental extensions of property rights to software developers. The Supreme Court has, in fact, referred to the separation of powers argument in justifying judicial reluctance to move decisively in the area.[148]

Finally, it is possible to argue that some value questions have been overlooked due to basic misunderstandings of the nature of computer technology. Professor Pamela Samuelson argues that the developing practice of extending copyright to the machine-readable form of computer software sacrifices the value of public disclosure of copyrighted works because the disclosure of programs in machine-readable form is not sufficiently instructive.[149] That is, copyright is granted even though meaningful disclosure has not taken place and such meaningful disclosure has historically been the *quid pro quo* of copyright protection. Samuelson sees this as a new problem because prior

146. *See* Brief for Computer Business Equipment Manufacturers Association at 20-22, Dann v. Johnston, 425 U.S. 219 (1976) (opposing patentability of software as harmful to innovation and rapid change); Brief for Amicus Curiae the Computer Business Equipment Manufacturers Association at 16-20, Gottschalk v. Benson, 409 U.S. 63 (1972) (patents will adversely affect future of industry); Brief for Amicus Curiae Burroughs Corp. at 9, Gottschalk v. Benson, 409 U.S. 63 (1972) (algorithms not patentable material); Brief for Amicus Curiae International Business Mach. Corp. at 19, Gottschalk v. Benson, 409 U.S. 63 (1972) (programs should not be patented).

147. "In this case the Government remains silent while the Computer Business Equipment Manufacturers Association distorts the history of the computer industry to assert that the 'computer industry has developed in an atmosphere of free interchange of computer programs.'" Brief for Amicus Curiae the Association of Data Processing Service Organizations and Software Industry Association at 3, Dann v. Johnston, 425 U.S. 219 (1976) (quoting Brief for Amicus Curiae the Computer Business Equipment Manufacturers Association at 10, Dann v. Johnston, 425 U.S. 219 (1976)).

148. *See* Gottschalk v. Benson, 409 U.S. 63, 73 (1972) (question of patentability raises considerable problems that only committees of Congress can manage).

149. *See* Samuelson, *supra* note 95, at 705-27. Professor Samuelson successfully explains that the "code book" cases are consistent with her thesis. *Id.* at 713-14.

to the advent of computer software in object code, it was difficult to imagine how a copyrightable work could be widely distributed in a useful form without disclosure.[150] Nevertheless, it is a problem which cries out for solution.

C. THE CAPABILITY PROBLEM: ABILITY OF LEGAL INSTITUTIONS TO APPLY SOFTWARE PROTECTION STANDARDS

Even assuming that the facts are understood, and that one has a clear conception of values, and hence goals, difficulties arise that reflect the limited capability of existing institutions to administer defined norms.[151] Capability problems exist in virtually all branches of the law of intellectual property, and these problems have significant ramifications for software protection.

Turning initially to patent law, the ability of the Patent Office to process applications on software has long been questioned. The Supreme Court in *Gottschalk v. Benson*[152] noted that the President's Commission on the Patent System had opposed extending patent protection to software, partly because it believed that the Patent Office lacked the resources and classification techniques necessary to process patent claims on software.[153] As the history of litigation on the question of patentability shows, the Patent Office has consistently opposed allowing claims on software dominated inventions.[154] One wonders whether this position has had more to do with the Office's concerns about practical limitations on its capacity to process claims than with the question of whether the claims fall within the bounds of patentable processes or devices. Some are

150. *See id.* at 710.

151. Some of the limitations on the capacity of the legal institutions are language related. Although the division of language-related problems into two portions (this section on capability and the next section on language) is an uneasy one, it is justifiable because the capability problems are arguably permanent while the language problems are capable of solution. The legal standard of unobviousness in patent law has proven to be difficult because it functions at an uncommonly high level of abstraction and experience will not likely lend much definition to its intrinsic vagueness. On the other hand, some language-related uncertainty may be relatively short-lived. Experience may eliminate language-bound problems like the problem mentioned in the watch hypothetical, *see supra* text at 538-40, because, as soon as the community of watchmakers adopts one language convention or another, their problem will disappear.

152. 409 U.S. 63 (1972).

153. *Id.* at 72 (citing President's Comm'n on the Patent System, *supra* note 39, at 13).

154. Chandler, *supra* note 1, at 230-53.

skeptical of the Patent Office's fears and believe that the Office could relatively easily meet the challenge of processing patent claims on software.[155]

The norms of patent law generally create problems in their administration because patent law is notorious for asking judges to apply criteria that are almost metaphysical in character. For example, no patent may issue in any field if the advance would be "obvious" in light of prior art.[156] Someone unfamiliar with patent law doctrine might note that, although a bit vague and perhaps subjective, this criteria seems no more metaphysical than "reasonableness"—a standard frequently encountered in many areas of law. But it turns out that what "nonobvious" means is more subtle than one might first imagine. First, imagine a person with ordinary skill in the art in question. If the invention is in the mechanical arts, for example, hypothesize a person with no special or extraordinary skills in those arts.[157] Then vest this ordinary mechanic with complete familiarity with all of the art in the field without, somehow, rendering the hypothetical mechanic extraordinary. Only then ask whether this hypothetical person would have found the advance to be obvious. This standard is so difficult to apply reliably that its use impairs the objectivity of standards in the patent field.

Furthermore, when patent law confronts claims involving software, the capability problems are magnified. Fact-finders will initially be quite unfamiliar with the art, making it difficult to construct the necessary hypothesis. Moreover, software may be so complex that the consideration of it in its totality may be, as a practical matter, impossible. As one author asked, "Can any court reasonably be asked to cope with an 'invention' two miles long?"[158] It is one thing to tell someone to consider something "as a whole," and quite another to actually do it. In one sense a program is like a machine, in that it is comprised of constituent parts, each of which functions in a predictable way,

155. "One computer science expert has stated that reviewing software patent applications for novelty, usefulness, and nonobviousness would not be difficult for most computer science experts." VANDERBILT NOTE, *supra* note 137, at 179 (citing Interview with Patrick C. Fisher, Chairman of the Computer Science Department, Vanderbilt University).

156. 35 U.S.C. § 103 (1982).

157. Note that this hypothesis is being formed in the mind of one who may be generally unfamiliar with the mechanical arts to begin with.

158. Gemignani, *supra* note 12, at 303. The author was referring to the SABRE flight reservations system used by American Airlines. *Id.* at 276 & n.36.

to contribute to the function of the whole. In another sense, however, a program is not at all like a machine, and it may be asking too much to suggest that one can visualize a program as one can a machine. The interactions between the parts may be more complex than the interaction of machine parts, and economy of code in software can sometimes create a complexity that even other skilled programmers may find difficult to follow.

Patent law's requirement of nonobviousness is not the only difficult task faced during the application of patent law to software dominated invention. Another is the utilization of a purported distinction between mathematical and non-mathematical algorithms.[159] The distinction arose in order to avoid the Supreme Court's holding in *Gottschalk* that a patent which would preempt a mathematical formula could not issue.[160] There has been some debate about whether the distinction is meaningful.[161] Some have suggested that the distinction is confusing and unnecessary[162] while others have characterized it as "absurd."[163] It may be based on an arguably naive and confounding distinction between formal symbol systems that use numbers and other formal symbol systems. Therefore, to the extent that a distinction lacks meaning in the eyes of those who will be required to implement it, any standard that relies on such a distinction will test, if not overcome, the capacity of judges, lawyers, and expert witnesses.

A non-software example from patent law may prove suggestive, and to some patent lawyers, I suspect, painful. In 1969, in *Anderson's-Black Rock, Inc. v. Pavement Salvage Co.*,[164] the Supreme Court ruled that a patent was invalid because it was based on an advance that was obvious. In the course of its opinion, the Court noted that there was no "synergistic effect" exhibited by the combination of elements.[165] In 1976, the Court in

159. *See* Paine, Webber, Jackson & Curtis, Inc. v. Merill Lynch, Pierce, Fenner & Smith, Inc., 564 F. Supp. 1358, 1366-68 (D. Del. 1983) (discussion and application of the distinction).

160. *See Gottschalk*, 409 U.S. at 72.

161. *See e.g.*, Davidson, *supra* note 1, at 641; Novick & Wallenstein, *The Algorithm and Computer Software Patentability: A Scientific View of A Legal Problem*, 7 RUTGERS J. COMPUTERS, TECH. & L. 313, 333-41 (1980); VANDERBILT NOTE, *supra* note 137, at 172.

162. *See* Gemignani, *Should Algorithms be Patentable*, 22 JURIMETRICS J. 326, 334 (1982).

163. *See* Samuelson, *supra* note 95, at 759 n.446.

164. 396 U.S. 57 (1969).

165. *Id.* at 61.

Sakraida v. Ag Pro Inc.[166] again alluded to the absence of synergistic effect in holding a patent invalid.[167] The combination of elements, according to the Court, did not "result in an effect greater than the sum of the several effects taken separately."[168] Since that time, a number of courts and scholars have struggled with the standard of synergism and many have expressly rejected it.[169] Some suggested that the reason for the difficulty is that synergism simply is a word that makes no sense when applied to an application for a patent on a mechanical device.[170] Others purport to find some meaning in it.[171] Whatever one believes, "synergism" has been a failure as a judicial standard. It has certainly caused a great deal of confusion; moreover, *even if* one makes the strong assumption that it is an appropriate standard by which to measure patentability, it appears to be a standard that courts are incapable of applying in a predictable and understandable manner. The distinction between mathematical and nonmathematical algorithms is arguably subject to the same infirmity.

Similar kinds of capability problems exist within the present copyright law as applied to software. The unfamiliarity of software posed, from the very outset, problems of capability. For example, CONTU had no computer scientists, representatives of the software or hardware industries, or users of sophisticated systems as members.[172] The commissioners struggled with the application of copyright principles, and arguably failed to reach an appropriate recommendation because, collectively, they never really understood the nature of software, especially as embodied in machine-readable form.[173] One of the funda-

166. 425 U.S. 273 (1976).

167. *Id.* at 282.

168. *Id.* at 282 (citing Anderson's-Black Rock Inc. v. Pavement Salvage Co., 396 U.S. 57, 61 (1969)).

169. For a judicial critique and rejection of synergism as a special standard of patentability for combination patents, see Rengo Co., Ltd. v. Molins Mach. Co., 657 F.2d 535, 543 (3d Cir. 1981), *cert. denied*, 454 U.S. 1055 (1982), and the cases cited therein. For a scholarly discussion and rejection of synergism, Crossan, see *Patent Law: Synergism Rejected*, 56 CHI. KENT L. REV. 339, 341-48 (1980).

170. *See* Republic Indus., Inc. v. Schlage Lock Co., 592 F.2d 963, 968-70 (7th Cir. 1979) ("[S]ynergism is only a figure of speech, for in its literal sense [it] never has existed and never can exist in mechanical or hydraulic inventions."); Crossan, *supra* note 169, at 341-48.

171. *See* Note, *Patentability of Mechanical Combinations: A Definition of Synergism*, 57 TEX. L. REV. 1043 (1979).

172. Samuelson, *supra* note 95, at 699 n.137.

173. *Id.* at 703-05.

mental distinctions serving to distinguish between the protectable and nonprotectable in copyright law is the distinction between idea and expression.[174] This distinction, difficult enough in cases of books and pictures, may be even more awkward as applied to software. In most cases in which copying is verbatim and complete, there is no necessity to draw a line between the protectable and the nonprotectable because some protectable interest must have been invaded. In a case, however, in which the language of a computer program, translated into electrical impulses, has functional characteristics, even literal copying may not free the court from the task of deciding whether the expression was coextensive with the idea.

The Third Circuit was forced to grapple with this problem in *Apple Computer, Inc. v. Franklin Computer Corp.*[175] The defendants had duplicated the operating system used on the Apple II computer.[176] When sued for infringement, they defended on the ground that because there were so few choices of software that would perform the necessary function of controlling disk drives even verbatim copying was privileged.[177] The defendants argued that the case was analogous to *Morrissey v. Proctor & Gamble*,[178] in which the First Circuit held that, because there is a privilege to borrow the idea, if there are very few possible expressions of that idea then an otherwise impermissibly similar expression will be immune from the charge that it was a copyright infringement.[179] In *Apple Computer*, however, the court decided only that, in that specific instance, other possible programs could have performed the same function, and therefore ruled for the plaintiffs.[180] If this limited choice theory can be a credible claim of privilege in a case of literal copying, one can only imagine how difficult it will be to obtain reliable and objective decisions in cases in which some changes have been introduced by copyists, and the borrowing is not a literal, line by line, recreation.

The idea-expression distinction can be seen from one perspective as a question of the copyrightability of the subject matter. From another perspective, it is a question of whether or not there was an infringement. Conceding that the plaintiff

174. *See* 1 M. NIMMER, *supra* note 83, § 2.03[D].
175. 714 F.2d 1240 (3d Cir. 1983), *cert. denied*, 464 U.S. 1033 (1984).
176. *Id.* at 1245.
177. *Id.* at 1253.
178. 379 F.2d 675 (1st Cir. 1967).
179. *Id.* at 678.
180. *Apple Computer*, 714 F.2d at 1252-54.

had a copyright, did defendant's borrowing violate the plaintiff's exclusive right? In cases of nonliteral copying that question will almost certainly be extraordinarily difficult. The question becomes whether the copied software is "substantially similar" to the original.[181] Even the process of comparison will be very difficult due to the length of many programs, the need for comparisons between object codes in different computer languages, and the relative unfamiliarity of judges and juries with even simple computer programs.[182]

Once again, the capability problems associated with semiconductor chips may prove, in comparison those associated with software, more manageable. Judgments concerning literal similarity at the final level of specificity of expression, the semiconductor chip, might be easier to make than those involving comparisons between software at higher levels of generality of expression. However, some capability problems remain. Many disputes concerning chips are likely to involve literal similarity,[183] and one of the more difficult questions that will likely arise under the Semiconductor Chip Protection Act in literal similarity cases is whether the chip is sufficiently "original" to meet the standard embodied in section 902(b). This originality standard is higher than that of the Copyright Act, and the judicial system therefore lacks experience in applying the standard.[184]

D. THE LANGUAGE PROBLEM: LEGAL TERMINOLOGY MEETS THE COMPUTER REVOLUTION

Perhaps the most intriguing of the problems contributing to uncertainty of the law of software protection is the problem

181. *See also* MINNESOTA NOTE *supra* note 1, at 1285-94 (arguing that the substantial similarity test involving an application of the ordinary observer stnadard is inappropriate and may stifle innovation if applied in the area of computer software).

182. *See* Whelan Assocs., Inc. v. Jaslow Dental Laboratory, Inc., 609 F. Supp. 1307, 1321 (E.D. Pa. 1985) (expression of idea embodied in a computer program entitled to copyright protection even though program must be altered and refined to be adaptable to different types of computers with different source codes). This decision is criticized in a column by Richard Stern. *See* Stern, *Opinion*, 5 EUROPEAN INTELL. PROP. REV. 123 (1985).

183. HOUSE REPORT, *supra* note 24, at 26, 1984 U.S. CODE CONG. & AD. NEWS at 5775; Wilson & LaBarre, *supra* note 99, at 91. It has been suggested that the determination of similarity may be *more* difficult because a purely visual standard will not be available. In the event that this is not true, the standard of similarity may be inordinately difficult to apply.

184. *See supra* note 105.

of language. The vocabulary of existing law is frequently inadequate to the challenge posed by software, and, if that were not enough, the language of software itself is not mature enough to provide the vocabulary for legal norms. Both kinds of problems promote uncertainty. Legal vocabulary often fails when employed in cases involving software, and the language of the software technician is flawed when used as the reference point for legal rules.

One example of the inadequacy of legal norms as applied to software is drawn from patent law and is the application of the so-called "mental steps" doctrine, i.e., no patentable process may include "mental steps."[185] Whether or not the standard is a wise one, in most cases in which it was utilized, it was applied within tolerable limits of uncertainty because only human brains were conceived as being capable of mental steps.[186] Early in the development of the argument over the patentability of software, it was observed that the computer does something remarkably like thinking and it was therefore argued that a process containing a computer therefore contained mental steps.[187] It appears now that the doctrine will not be applied to software, even though what a computer does is in some respects analogous to thinking.[188] In fact, some artificial intelligence researchers would probably argue it is fundamentally the same,[189] but that discussion is beyond the scope of this Article. The mental steps debate, however, is fairly typical of the kind of problem that arises when a new phenomenon surfaces which cannot comfortably be accommodated within the pre-existing vocabulary.

We have seen, for example, the difficulty in characterizing computer software as a process or a device.[190] The difficulty arises because of the extraordinary capacity of software to bridge the gap that normally exists between the description of the thing and the thing itself. On one hand most computer

185. *See, e.g., In re* Abrams, 188 F.2d 165, 166 (C.C.P.A. 1951) (Counsel for appellant suggested a rule of law stating that "If all the steps of a method claim are purely mental in character, the subject matter thereof is not patentable within the meaning of the patent statutes.").

186. *Abrams*, 188 F.2d at 169-70.

187. The Patent Office in *In re* Prater, 415 F.2d 1393 (C.C.P.A. 1968) had taken the position that a digital computer contained mental steps; the CCPA reversed holding that there was no issue concerning "mental steps" with regard to the item sought to be patented. *Id.* at 1406.

188. *In re* Musgrave, 431 F.2d 882, 888-89 (C.C.P.A. 1970).

189. *See* D. HOFTSTADTER, *supra* note 11, at 594-680.

190. *See supra* notes 45-60.

software in its written, source code, form seems to describe a process for manipulating information or symbols.[191] A programmer can, in the course of reading from the printed program, step through the process of manipulation; this is in fact a very common way to find errors in programs.[192] On the other hand, a programmer sometimes begins to work with the program in its entered form, in the state in which it is more nearly analogous to a set of switches.[193] A programmer may "peek" at a location in the computer's memory registers, and in that way determine the state of the operation at that moment and whether some manipulation is required.[194] It must be remembered that much of the time the entire program "resides" in the computer's memory, like a machine, and at any particular moment only some parts of it are "working." The program may also modify itself as it operates, changing values of variables, and reshaping itself in other ways. The program may have, on one hand, a static aspect and yet, on a command, become a dynamic, changing thing. A particular program may, in some applications, be "burned in" so that it has a permanence that makes it more useful and usually faster. In fact, it would be theoretically possible to build a hard-wired version of virtually any program, or at least of substantial portions of most programs.[195] It is therefore open to question whether the program is more like a wiring diagram, and so a machine, or more like a description of a process, implemented on a machine. It appears that, because of the uncertainty about the balance between hardware and software in any given case, it has become useful for computer scientists to speak of the hardware *plus* the software in their interactive state as the "virtual machine" or the "virtual computer."[196]

The pre-existing distinction between machine and process did not need to be sharp before the software challenge arose. Language is shaped by the functions it serves, and it is therefore not surprising that no pre-existing distinction was ready to

191. *See supra* notes 17-19 and accompanying text.

192. Although more complex programs generally rely on error trapping routines, stepping through the program remains an important way to find and correct problems in the program.

193. *See supra* notes 20-22 and accompanying text.

194. *See, e.g.*, L. POOLE, THE APPLE IIC USER'S GUIDE 133, 393-94, 427-33 (1985).

195. *See supra* note 22.

196. Samuelson, *supra* note 95, at 680 n.61.

be taken from the shelf, to be used in light of the new phenomenon.

Another possibility exists, namely that a computer program is a computer program and is neither process nor machine. Efforts to capture the essence of a program within the dichotomy of process or machine may be ill-conceived. It has been argued that computer programs in machine-readable form should be protected, if at all, by a new law specifically designed to deal with the special problems of such creations.[197] Once again the comparison with semiconductor chips is instructive. The Semiconductor Chip Protection Act constitutes a recognition, in part at least, that pre-existing categories simply were inadequate to deal with a new technology. Because the "fit" of the old rule to an utterly new phenomenon was so imperfect, it was necessary to create a new form of protection that was designed with the new technology in mind.[198] Although it might be tempting to believe a similar solution would be appropriate for software, such a solution might be more difficult to achieve because the diversity of software itself, as compared to semiconductor chips, poses a different kind of problem. That is, semiconductor chips are, it seems, a more narrowly defined class of creations and thus better suited to a special form of protection.

In a 1908 copyright case, which may have been the first software case, the Supreme Court was confronted with the question of whether a piano roll was a copy of a musical work.[199] The Court, perhaps due to a conceptual blind spot, ruled that the piano roll was not a copy, a decision that created headaches for copyright lawyers, songwriters, record companies, and software proprietors for more than six decades. The Court could not accept the proposition that the embodiment of information in a form not directly perceptible by humans could be a copy. The pre-existing formulation of the standard proved inadequate in the face of a new kind of embodiment of an intellectual creation. Similarly, in 1979, in one of the last cases decided under the 1909 copyright statute, a federal court ruled that a computer program stored on a ROM chip could lawfully be copied because in that form it was not a copy of the writing that was the program, just as the information stored on the pi-

197. *See* Samuelson, *supra* note 95, at 663, 671.
198. *See* Samuelson, *supra* note 114, at 486-501.
199. White-Smith Music Co. v. Apollo Co., 209 U.S. 1, 18 (1908).

ano roll was not a copy of the song.[200] Some might argue that in these cases the problem was really not a problem of vocabulary, for one does not need to violate the boundaries of a sensible definition of "copy" in order to include a device that stores information in a format not immediately accessible to humans, but rather a problem of imagination and conceptualization when faced with a new technology. Others might find that the utilitarian nature of the piano roll, which plays an active role in the mechanical operation of the piano, is enough to distinguish it from other, passive, information storage devices.

Another example drawn from copyright law relates to the copyright truism that copyright does not entitle one to control the art that is the subject of a copyrighted explanation, even in a case in which the art described is the original creation of the author.[201] Even though one may copyright a set of building plans, that copyright does not prevent another from constructing the building revealed in those plans.[202] One may copyright a book describing a bookkeeping system, but that confers no control over the use or exploitation of the system itself.[203] It could be asserted that this principle should be applied to software, and that the source code version of a computer program is to the object code version as the architect's plans are to the building.[204] Some commentators, however, have expressly rejected what they call the "fallacy" of this analogy.[205] The acceptance of the analogy would certainly render copyright protection for programs incomplete because, by copying the object code version of the program, a pirate could escape the purview of copyright protection. The pirate would not, of course, be entitled to recreate the source code, though that would likely be unnecessary once the pirate had copied the object code version. Once again, the meaning of a legal standard is uncertain in the context of an entirely new phenomenon.

The second of the language related dilemmas is the reverse aspect of this problem. Existing legal standards may be expressed in language that is of uncertain value when used to ar-

200. Data Cash Sys. v. JS&A Group, Inc., 480 F. Supp. 1063, 1068 (N.D. Ill. 1979), *aff'd on other grounds*, 628 F.2d 1038 (7th Cir. 1980).

201. Baker v. Selden, 101 U.S. 99, 102 (1879).

202. Scholz Homes, Inc. v. Maddox, 379 F.2d 84, 86-87 (6th Cir. 1967).

203. Baker v. Selden, 101 U.S. at 102.

204. This is, essentially, the action the district court took in Apple Computer, Inc. v. Franklin Computer Corp., 545 F. Supp. 812, 823 (E.D. Pa. 1982), *rev'd*, 714 F.2d 1240 (3d Cir. 1983), *cert. denied*, 464 U.S. 1033 (1984).

205. *See* Davidson, *supra* note 1, at 630.

ticulate that standard for application to a new technology. Moreover, because legal standards rely on the existence of meaningful categorization of the phenomenon subject to regulation, it is necessary to find a way to define the boundaries of those categories. Usually there is an adequate existing vocabulary, whether it is comprised of words of ordinary meaning or a specialized language utilized by those in the trade. This vocabulary can be used to capture the distinctions that the law uses to mark the in and the out, the permitted and the forbidden. Computer technology, however, has been advancing so quickly that there may be no stable body of distinctions and associated terms that can be adopted for the purposes of standard setting.[206] Technicians develop a language, as they feel the need, in order to serve their own objectives. In a fast moving field, there may be no term which adequately captures the distinction that legal policy demands. Even if, for a time, there seems to be such a term, it may undergo its own kind of technological obsolescence and come to mean something else, or in fact become meaningless.[207]

Perhaps the most dramatic example of the instability of language in the interaction of software and the law of intellectual property is the dissolution of the distinction between hardware and software. Early legal argument seemed to make

206. [T]he function provided by software . . . does not even have, at this point, a useful taxonomy." Bacon, *supra* note 28, at 775. The author later notes that, "[s]ince the work that computers do will be driven by the virtually uncountable numbers of tasks that humans may want to undertake, one begins to see why it is so difficult to develop a taxonomy of computer software at any but the most primitive logical levels." *Id.* at 777.

207. A vivid example of this outside the software field can be found in the criminal law. The medical specialty dealing with mental disorders has developed a vocabulary designed to capture medically significant distinctions. Some of those concepts and the related vocabulary were borrowed from psychiatry and proved influential in the development of the concept of responsibility in the criminal law. Psychiatrists were, and remain, frequent participants in trials in which the issue is the responsibility of an accused, and the language of the rapidly changing medical field is often employed in the application of the legal norms. Only recently have doctors and others suggested that the terminology within which the medical opinions were given is a terminology reflecting an underlying set of criteria relevant to the medical question of treatment, but not necessarily relevant to the legal or moral question of responsibility. That is, the question of whether a person was in need of treatment differs from the question of whether it is appropriate to incarcerate that same person in order to serve the purposes of the criminal law. *See* M. FOUCAULT, DISCIPLINE AND PUNISH: THE BIRTH OF THE PRISON 19-22 (1977); M. FOUCAULT, MADNESS AND CIVILIZATION: A HISTORY OF INSANITY IN THE AGE OF REASON 221-40 (1965).

much of this distinction, and seems to continue to make much of it at a time when it is becoming increasingly less significant within computer technology. The concept of "firmware" has been introduced, and those familiar with the field have begun to realize that, to some extent, the hardware-software distinction is not a categorical distinction, but rather names two ends of a continuum that refers to the form of storage of a digital logic system.[208] It seems that in many cases the question of whether a particular advance will be embodied in hardware or software is a matter of choice, a choice driven by the function to be played by the programmed unit, rather than some inherent difference between hardware and software.[209] Not only may doctrinal language be meaningless when applied to new phenomena, and words fail to make some legal, as opposed to technical distinctions, but errors can creep into the analysis through the use of words to which we do not attach technical significance. It seems that CONTU made just such an error in the use of the word "read" in its report.[210] The word "read", as most of us use it, has to do with the process of human interpretation of printed language. We all know what it means to read something. Computer scientists also use the word "read", but they mean something quite different. "Read," when used by computer scientists, describes a process of retrieving or loading signals electronically and is only in some respects analogous to our use of "read" in everyday speech. It is critical to realize that a great deal of the language of computers is language by analogy. Available words are used as metaphors and can easily be misunderstood if interpreted literally.

CONCLUSION

What does all of the preceding mean? It has not been my intention to confuse, though I may on occasion have appeared to transform simplicity into complexity. My purpose has been

208. *See* Sprowl, *supra* note 15, at 785-98.

209. The forces which affect this choice are usefully explored in Stern, *MicroLAW*, IEEE MICRO, Feb. 1983, at 74-75. The author speculates that one possible factor influencing the decision whether to embody an advance in software instead of in hardware is the perceived legal bias against hardware. "The law seemed to be more willing to protect the earlier software . . . against copying by competitors than the later, more efficient hardware." He concludes that passage of the Semiconductor Chip Protection Act may have "counteracted the artificial bias that the legal system was imposing on product engineering." *Id.*

210. Samuelson, *supra* note 95, at 724.

simply to explain what I believe to be the sources of some of the uncertainty and confusion that has surrounded efforts to grapple with legal protection for computer software and semiconductor chips and to suggest that, in the absence of *sui generis* protection legislation for software much like that directed to semiconductor chips,[211] only patience may be appropriate. To the extent that the meaningful application of rules requires understanding of the facts, it may be some time before the legal profession, including the judiciary, and other organs of government can develop that mastery. To the extent that there is no consensus as to the objectives that the rules are intended to serve, it is not surprising that the rules themselves are uncertain. To the extent that the rules are to be sensible, they must embody administrable distinctions, and the agencies asked to apply the rules must have resources adequate to the task. Finally, the law can scarcely be expected to provide clear rules until it has the language tools which are prerequisites to drawing the distinctions all rules require. Those distinctions must make sense as a matter of legal policy, but they also must be based on distinctions that are meaningful within the technology itself. To the extent that the technological language is new, or changing, or inadequate, the law will necessarily mirror those inadequacies. It is important to avoid doing, in the short run, anything that will prevent us from doing, in the long run, what is sensible.

211. Professor Samuelson argues for just such a development, to be initiated by another commission, much like CONTU, but dominated by computer scientists who are more likely to understand the technology than were the members of CONTU. Samuelson, *supra* note 95, at 762-69.

Richard Stern has offered a provocative suggestion for *sui generis* software protection. *See* Stern, *MicroLAW*, IEEE MICRO, Apr. 1984, at 69-70; Oct. 1983, at 49-52; Aug. 1983, at 88-91; June 1983, at 62-64.