THE ECONOMICS OF INTELLECTUAL PROPERTY PROTECTION FOR SOFTWARE:

The Proper Role for Copyright*

Kenneth C. Baseman†
Frederick R. Warren-Boulton†
Glenn A. Woroch§

April 1995

Abstract

This paper provides an economic analysis of the proper scope of copyright protection for computer software. We begin by identifying key economic characteristics in the production and use of software; notably, the costs to developers is largely fixed and sunk, users often incur substantial sunk costs, and that the value of software to users is usually a significantly increasing function of the total number of users (*i.e.*, "network externalities" are important).

We then use economic theory and analysis to establish three propositions. First, we demonstrate that the copyright protection granted to the original developer of a software package should not extend to elements of the software that achieve the status of a *de facto* standard because the resulting monopoly leads to pricing that fails to achieve efficient dissemination of the software and fails to reward other sponsors who have invested in the *de facto* standard. Next, we argue that software interface specifications also should not be copyrightable since it would permit an inefficient extension of market power to complementary software and to later improvements. Finally, we favor reverse engineering for the purpose of achieving interoperability since it enables firms to efficiently design compatible programs and to guard against unwarranted abuse of copyright protection.

In most instances, recent case law is consistent with these principles, especially since the recent Appeals Court decision in Lotus v. Borland. Importantly, copyright law has devised a "merger doctrine" that denies protection whenever a product is the (nearly) unique expression of an uncopyrightable idea, a principle that effectively implements our prescriptions for software copyright. Since we conclude that copyright is the appropriate form of protection for intellectual property only when the likelihood of an unwarranted grant of monopoly is extremely low, this prescription achieves the desired balance between the need to reward innovative developers of software programs and the need to encourage suppliers of complementary products and those who build upon and advance prior work.

---

* - We are grateful to Peter Choy, Russ Wayman, and Christine A. Owens for their comments on an earlier draft of this paper. We also thank John Barton and Yale Braunstein for useful discussions.
†- Microeconomics Consulting & Research Associates, 1875 I Street, N.W., Washington, D.C. 20006.
§ - Department of Economics, University of California-Berkeley, Berkeley, CA 94720.

I. PROTECTING INTELLECTUAL PROPERTY IN SOFTWARE: HOW MUCH IS TOO MUCH?

Determining the optimal form and degree of protection for intellectual property poses special problems. Private sale of information encounters several inherent difficulties. In particular, it is difficult to communicate the value of information without revealing that information first, at which time the potential buyer has acquired the information at no cost.[1] Governmental intervention is clearly desirable to establish property rights in information and to prevent users from "free riding" inappropriately on the efforts of its creators. To determine the optimal extent of those property rights, however, the policy maker in search of efficiency faces a dilemma caused by another peculiar feature of intellectual property: it displays a characteristic of a "public good" in that it is costly to produce but costless to use. Consequently, efficiency in production requires that the producer receives a positive price, whereas efficiency in distribution requires that users should pay a zero price.

It comes as no surprise, therefore, that reasonable observers differ vastly in their views regarding the appropriate level of protection for intellectual property. Those who focus on the free-rider problem allege that producers are often unable to appropriate even a small proportion of the value of their efforts. Typically, they perceive attempts to limit protection as simply attempts by less competent competitors to handicap their more ingenious rivals, leading them to conclude that intellectual property is unlikely ever to be overprotected in practice.[2]

On the other side of the issue are those who contend that few discoveries are made in isolation, and that most scientific advances build on the contributions of others, many of which are in the public domain.[3] A breakthrough in basic research in a related field, or spillovers from more distant technical areas, can greatly reduce the cost of developing specific applications. Allowing the developer unrestricted rights to those marketable applications allows the grantee to profit from prior work in excess of the grantee's actual contribution. In addition to any equity considerations, such overprotection is inefficient: it results in excessively high prices that, in turn, cause underutilization of that information.[4] Equally important, it can lead to wasteful competition to gain those rights that dissipates much of the value of the underlying prior work. This occurs when potential grantees devote excessive resources to winning a "patent race."

This paper examines one particular kind of intellectual property protection: copyright protection for computer software. Our analysis does not attempt to determine the "optimal" level of such protection,[5] but instead seeks appropriate bounds on the level of that protection. Our criterion for overprotection is whether copyright protection of a piece of software leaves the rest of society worse-off; *i.e.*, whether the copyright holder receives more in monopoly rents than the copyrighted software has added to total welfare.[6] For intellectual property,

---

[1] See Arrow (1962).

[2] For example, see Miller (1993) and Judge Keeton's decision in Lotus v. Paperback.

[3] See Barzel (1968) and Landes and Posner (1989).

[4] Unless, of course, the seller was able to perfectly price discriminate among users, an impossible task given the asymmetry of information.

[5] Such an exercise would have to address the public-good nature of information. Specially, it would require some practical way of ensuring that producers of information recover an amount only slightly in excess of its cost, assuming that cost is less that the social value of the information. This rule has been proposed by Landes and Posner (1989) and Menell (1987, 1989).

[6] More precisely, the *ex ante* expected value of the reward to a winner of the copyright should be related to the *ex ante* expected value of the winner's incremental contribution over and above the surplus that would have been generated if the other contestants had competed in

overprotection would be possible if, perhaps because of prior work, the discovery would have been made soon anyway without any protection whatsoever, or with much less protection and thus at lower cost to the rest of society. Intellectual property protection would also be excessive if the grantee could raise the costs and reduce the opportunities available to suppliers of substitute products.

Copyright law attempts to prevent such leveraging, however, by making the distinction between "idea" and "expression." An expression can operationally be defined as a discovery or work that would not otherwise have been made (*e.g.*, no one besides Mary Shelley would have written Frankenstein), and protection for which would not reduce the opportunities available to any other author. In contrast, granting exclusive rights to an idea (*e.g.*, a novel about the creation of a humanoid monster) would significantly limit the alternatives available to other authors and would allow the copyright holder to appropriate the value of something she did not create.

To determine the efficient bounds on the application of software copyright, however, we need to introduce two concepts from economics. The first is the notion of "network externalities." Network externalities occur when the value of a product or service increases with the cumulative number of users. When this is the case, each additional purchase raises the value to existing users as well as the expected value to future adopters.

The second economics concept follows from a simple expected-value calculation.[7] We argue that the uncritical, automatic and extensive nature of protection under copyright is efficient only when the expected value of the welfare loss of its being granted mistakenly is *de minimis*. Like summary judgment and the *per se* rule, copyright is the cost-effective approach only when the probability of error (*i.e.*, overprotection) is very small, and/or when the potential social loss associated with the error is very small.

As we argue below, allowing the copyright of interface specifications or of expressions that become *de facto* standards can violate our criterion for overprotection (*i.e.*, can make the rest of society worse off) when network externalities are present.[8] In that event, the copyright holder may be able to appropriate the result of the efforts of others, and to raise the costs or reduce the opportunities available to suppliers of substitute products.

A similar concern arises where copyright of interface specifications would permit the copyright holder to control the supply of compatible complementary products. At best, the welfare effects of allowing such "vertical control" are ambiguous, implying that at least some justification should be offered before the state automatically grants property rights that would confer such control.

This does not necessarily mean that innovators should be denied rights to software that has these properties. Where the innovator has made a substantial contribution, property rights may be available through patent or trademark protection. Interface specifications and standard program elements should not, however, receive the extensive protection that is automatically and uncritically offered by copyright.

II. THE ECONOMIC CONDITIONS OF

---

the absence of the actual winner.

[7] "Expected value" is the sum of the probability of each outcome multiplied by the value of that outcome. *E.g.*, the expected value of a ¼ chance of $10 and a ¾ chance of $20 is equal to $17.50.

[8] As discussed below in Section III.A., many elements of a software program embodying a *de facto* standard will remain eligible for copyright protection, and the whole program may also receive protection. Our objection is to extending copyright to protect the elements of the program necessary to conform to, or practice, the standard.

THE SOFTWARE MARKET

A. The Nature of the Cost Conditions of Computer Software Development and Distribution

The production and distribution of computer software share several characteristics with other forms of intellectual property. First, most of the cost of developing a program (*i.e.*, writing, testing and debugging code) is independent of the number of copies produced. Marketing a software package (*i.e.*, advertising and distribution) may enjoy significant scale economies as well. And since the cost of duplicating and shipping the programs is negligible, once it is developed, the marginal cost of software is negligible.

A second distinguishing feature of software production is that much of these costs is sunk: a large fraction of development and marketing expenses cannot be recovered should the vendor decide to exit the business. The code usually has little value in other uses, and any learning acquired in the process can only partially be applied to other endeavors.

A third feature is that software developers are not the only ones who make substantial investments in software products. Besides out-of-pocket expenses for the software package itself and for ancillary hardware and software needed to run it, users make considerable intangible investments. They acquire expertise while learning and operating the program, and they create files and programs that are specific to the software package. These assets are made worthless if the vendor creates a new version of its software having specifications that are incompatible with the old version. Makers of compatible hardware components and software programs could find themselves in the same predicament. Expenditures by all these groups will be less sunk, in general, when industry technical standards ensure that their components are interoperable.

A final common feature is that the cost of developing a software package also depends on the stock of technologies that are technically and legally available to today's programmers. Developers of future generations of software benefit from the insights, as well as the mistakes, of current and earlier programmers. These benefits may derive from breakthroughs, such as object-oriented programming, that promise to improve the performance of all kinds of software, or advances in software that perform a narrow set of computing tasks. In either event, artificial restrictions on the use of past discoveries will necessarily raise the cost of current development. Restrictions on the use of general knowledge by subsequent innovators are particularly damaging to the social welfare and should be avoided.

B. Network Externalities

A full understanding of the software market must take account of the presence of network externalities. Network externalities occur when the value of a product or service to a buyer increases with the cumulative number of other buyers.[9] When this is the case, each additional purchase raises the value to current users, and eventually to future purchasers.

The clearest example of a network externality is the telephone network. At one extreme, owning a phone has no value if you are the only person connected to the network. Telephone service becomes more valuable to each subscriber as more households are connected to the network. In the case of software, there are several reasons why purchase of a software package delivers more value if many others own and use that same program. First, each user has more possibilities to share files and exchange expertise with other users. Second, a larger customer base can support production of a greater variety of complementary hardware and software by allowing recovery of the fixed costs

[9] The literature on network externalities has grown rapidly in recent years. Among the important articles are Katz and Shapiro (1985, 1992) and Farrell and Saloner (1985). Gilbert (1992) provides a nice overview of the literature. Menell (1987, 1989) criticizes copyright doctrine for ignoring the importance of network externalities.

associated with the development of these products. Since users benefit from hardware components and software applications used in conjunction with a software package, a person's willingness to invest in a hardware or software system will depend directly upon the cost and variety of such complementary products.[10]

The user need not depend on one manufacturer to supply this variety, however. As long as products made by different vendors are compatible, users can mix-and-match components and create new hybrid systems better suited to their personal tastes.[11] For example, when evaluating whether to buy a particular home video game system, consumers consider the cost and variety of all compatible game cartridges, whether originally designed for that system or for some other system.

The large social value of compatibility among software programs is revealed in vendors' design strategies. They go to great lengths to ensure that upgrades are backward compatible with earlier versions. More recently, software developers have marketed "suites" that contain an array of programs (*e.g.*, a word processor, a spreadsheet, and a database manager), and boast the ability to interchange files among the different applications. Presently, initiatives are underway to develop personal computer operating systems that will permit interchange of files among programs supplied by different vendors.

III. AN ECONOMIC ANALYSIS OF COPYRIGHT PROTECTION FOR SOFTWARE

The economic conditions of the software market—massive scale economies in the form of large fixed, sunk costs of development and marketing plus extensive network externalities—have important implications for the efficient form and extent of intellectual property protection. Network externalities in particular lead us to recommend severely restricted use of copyright for software. First, when software programs achieve the status of a *de facto* standard, copyright protection allows the possibility of leveraging the monopoly into complementary hardware and software. Second, the same analysis applies to software interfaces used by popular programs. Finally, reverse engineering can realize efficiencies of incorporating advances into the next generation of software and provide a check on monopoly power that comes from first-mover advantages or from excessive intellectual property protection.

A. Copyright Should Not Be Extended to *De Facto* Standards[12]

Creation of a *de facto* standard is a joint undertaking. It requires a first adopter, then a second, and a third, and so on. By itself, no hardware or software innovator can achieve market dominance by mere commercial launch of a product, no matter how early it arrives in the market (*e.g.*, *VisiCalc*) or how large its sponsor may be (*e.g.*, IBM's microchannel). A software package attains the status of *de facto* standard through the efforts of many sponsors. Besides the original developer, there are the users that purchase the program, the makers of complementary hardware and software, and even suppliers of compatible substitutes.

The multiplicity of sponsors of a *de facto* standard is a product of network externalities. We saw above how each additional purchase enhances the value of the program to other current users. Each purchaser also contributes to the standard's popularity by increasing its potential value for

[10] The courts have recognized the presence of network effects for personal computer operating system software. See Apple Computer v. Microsoft Corp., 717 F.Supp. 1431.

[11] The implications of this possibility are explored in Matutes and Regibeau (1988).

[12] Here we are talking about the standardized elements of the program which are necessary to practice, or conform to, the standard. This does not include those elements of the program that are not essential to practice the standard, nor do we mean to imply that the program as a whole is unprotectable.

future consumers because a larger customer base will also raise profit expectations of makers of compatible hardware and software products whose expansion, in turn, further enhances the standard's use value.[13]

Even firms that offer competing products may contribute to the success of a standard. While these products may vie for the same customers as the dominant product, they also attract users who would otherwise opt for an alternative platform or who would not make any purchase at all. When users of alternative products consider upgrading, they become prime candidates for the purchase of the dominant product. By that time, they have made sunk investments in training and in complementary hardware and software that significantly lowers the cost of switching to the dominant product—provided of course that the two programs are technically compatible.

As previously explained, each sponsor of the *de facto* standard—users of computer software, and firms that supply compatible hardware and software products—make considerable investments in the *de facto* standard. They also all bear the risks of its failure. The salvage value of these investments can be negligible if the prospective standard should fail to achieve critical mass, or if its specifications change along the way.[14]

For efficiency's sake, each adopter should receive a return equal to the incremental benefits that its purchase confers on others over time. The *n*-th purchaser should have claim to the incremental benefits that its purchase provides for the current *n-1* users through the remainder of the product's useful life. A purchaser should also have claim to the stream of incremental benefits enjoyed by the *n+1*-st purchaser, the *n+2*-nd purchaser, and so on.[15] Furthermore, it is possible that, by forgoing a purchase, a user raises the chance that other users will abandon the product, possibly switching to a more promising alternative.[16]

In principle, a *de facto* standard's original developer may find it profitable to "internalize" the incremental benefits of the many sponsors through the use of "penetration pricing." By setting lower prices to initial adopters, the innovator compensates them for the benefits they provide later adopters and for the risk they bear that the product will not achieve critical mass, and even possibly for the risk that the original developer will eventually exploit their locked-in position. Often, software vendors vary prices across customers and over time partly to reflect costs of serving them as well as their willingness to pay. They are also known to adjust licensing terms over time to vendors of complementary hardware and software.[17]

In practice, however, a firm granted ownership over a *de facto* standard may find it profitable to set prices well above marginal cost, which will inefficiently discourage use. To make matters worse, the monopoly sponsor will be

---

[13] Also, while a purchase benefits current and future users, it may hurt users of a competing standard which, in the limit, could be abandoned, stranding all of its adopters.

[14] In response, these firms may take costly measures to reduce the risk associated with adhering to the monopoly-sponsored standard; *e.g.*, they may design their products (at an additional cost) to ensure they can be "ported" to different hardware or operating system platforms.

[15] This conceptual experiment envisions purchasers who exclusively occupy a position in the sequence of purchases. Should a user forego a purchase, and if another user automatically enters and makes the purchase instead, then the incremental benefit of a particular user's purchase is significantly reduced. This would be the case, for instance, if there was an unlimited number of possible consumers.

[16] While, in theory, these direct benefits could be measured, it could be particularly difficult, for example, to estimate benefits from heightened expectations that the prospective standard will exceed some critical mass.

[17] For example, home video game manufacturers are reported to introduce new hardware platforms with especially attractive licensing terms to game developers. See Sheff (1994).

tempted to charge initial customers higher prices to extract a greater portion of their higher willingness to pay. The tendency to "price skim" (*i.e.*, price discrimination over time) leads a monopolist to set prices that fall, not rise, over time.[18] Lastly, information costs impose severe limits on the degree to which the monopolist can price discriminate across buyers based on their incremental contribution to network benefits.

Given the price mechanism's limited ability to support efficient adoption incentives when network externalities are present, assignment of copyright can greatly overcompensate the original creator of a *de facto* standard. Moreover, monopoly rights may be given to the innovator of what is later revealed to be an inferior technology. This happens with surprising frequency even without government intervention,[19] and copyright protection for *de facto* standards will raise that likelihood.

B. Interface Specifications Should Not Be Copyrightable

Misassignment of copyright can be especially pernicious in the case of interface specifications. To understand why, it is necessary to refer to the copyright doctrine of merger. The "merger doctrine" holds that when idea and expression are one in the same, or when there are only a small number of expressions of an idea, the expression is not protectable under copyright. In that event, control over expression would entail control over an idea. In economic terms, copyright of an expression would convey unwarranted monopoly power over the idea. Moreover, control of the initial idea may prevent, or limit the ability of, others to build on the idea, allowing an extension of the monopoly to other complementary products or to later improvements on the idea.

It is widely agreed that this basic notion of copyright--that expression should not be protected under copyright when idea and expression merge—does and should apply to software. In a network context, the same analysis implies that, if a program achieves the status of a *de facto* standard, then the courts should not rule out a merger of idea and expression simply because other, now inferior, expressions of the same idea were, and still are, available. To see this, consider the following not-so-hypothetical example.

Suppose that *ex ante* there are numerous ways to efficiently compose a piece of software, such as a spreadsheet, and that several companies in fact develop a spreadsheet program. Consumers have very strong preferences for standardization of software offerings, so that any spreadsheet package that obtains a lead in installed base quickly becomes the uniform selection of subsequent consumers. Then, *ex post*, any innovative and improved version of a spreadsheet must be compatible with the established standard.[20]

If the copyright covers the interface specifications with which the spreadsheet interacts with other software, with hardware, and with users, then the copyright on the initial standard essentially forecloses competition for future spreadsheet generations. Note that no social purpose is achieved by this grant of a valuable copyright monopoly.[21] Any of the spreadsheets configurations would have been equally valuable as the standard since *ex ante* there were numerous ways to express the idea(s)

---

[18] Cabral, Salant and Woroch (1994) show that in most cases a monopoly provider of a product that confers network externalities will never set real prices that increase over time.

[19] See Arthur (1989) and David (1985).

[20] *Ex ante* refers to the period before sunk costs are incurred. *Ex post* refers to the period after sunk costs have been incurred.

[21] The analogy here is with the monopoly power that economies of scale can confer upon pipelines or electric utilities. The usual policy response to massive economies of scale in production has been utility regulation to prevent the exercise of monopoly power. In computer software, where network effects correspond to economies of scale on the demand side, limits on the extent of copyright protection are all that is necessary to prevent the exercise of monopoly power.

embodied in the spreadsheet. In that case the creation of a copyright monopoly that covers interfaces is not in society's interest.[22]

More generally, since copyright should provide protection only in circumstances where there is a low probability that an unwarranted monopoly will have been awarded (*i.e.*, a small chance of a "false positive"), and since control over interfaces can easily convey market power and control over substitutes and complements, interface specifications should not receive the automatic and uncritical protection afforded by copyright.[23]

C. Reverse Engineering Should Be Allowed

Much of the recent controversy concerning software copyrights has involved reverse engineering. Various methods of reverse engineering (including so-called black box techniques, disassembly, and decompilation) are apparently widely used for a variety of purposes by customers, designers of complementary applications, and direct competitors.[24] Reverse engineering to develop directly competing products is the most controversial. There are several reasons, however, why reverse engineering for certain purposes should be allowed, even by suppliers of competing products.

First, traditional copyright analysis implies that reverse engineering to achieve interoperability should be lawful in markets with network externalities. Let us continue with the hypothetical example from the previous section, and now suppose that the interface standards are not covered by copyright, but that the developer of the initial spreadsheet chooses not to make all interfaces public. We retain the assumption that users invest heavily in their spreadsheet applications, and would be unwilling to switch to another spreadsheet if it meant losing the network advantages of staying with the standard. Then companies that seek to compete for the new generations or improved versions of the spreadsheet must deduce the interfaces and internal operations of the dominant spreadsheet (or license the software at monopoly rates).

Arguably the most cost effective means of doing so involves making a "copy" (via decompilation or disassembly) of the standard spreadsheet. Should would-be competitors be prevented by copyright from doing so? The answer is no. *Ex post*, it is consumer actions and not the inherent superiority of the initial spreadsheet that has created a merger of expression and idea since, as a practical matter, second generation spreadsheets must be fully compatible with the first generation standard to appeal to users.

Copyright law should allow such reverse engineering either on the grounds of an *ex post* merger of expression and idea, or on the grounds that these methods of reverse engineering merely allow competitors to (perhaps only partially) restore the *ex ante* circumstances where in fact there is no

---

[22] Menell (1987, 1989) shares our concern that copyright protection should not extend to interfaces, at least for software achieving the status of a *de facto* standard. Unlike us, he has reservations as to whether the correct result can be reached applying copyright's merger doctrine, and he suggests compulsory licensing, applying the fair use doctrine to limit the protection obtainable for interfaces, or creating a new category of intellectual property for software, as other means of getting to the right answer.

[23] Copyright protection is automatic in the sense that its grant occurs at the request of the developer. Copyright protection is uncritical in that application is not reviewed according to novelty or nonobviousness as in the case of patents.

[24] At one extreme is literal duplication of the object code. At the other is decompiliation for the purpose of academic research. Often entrants may reverse engineer programs in order to assure compatibility with complementary applications or for future compatibility with successive versions of the program. They may seek access to non-copyrighted potions of the program. Users may also disassemble a program so as to customize its operation to conform to their special needs or operating environment.

advantage to the first spreadsheet.[25] That is, the initial spreadsheet standard cannot expect to have it both ways—arguing that the copyright is valid because *ex ante* there were many commercially viable ways of expressing the idea, and then using copyright to prevent other competitors, who are faced with an *ex post* merger of idea and expression, from seeking to restore via reverse engineering the *ex ante* circumstances that made the copyright valid.[26]

A second reason to allow reverse engineering is that markets may be biased against open systems.[27] This is because in network markets proprietary systems may have a strategic advantage unrelated to efficiency. Proprietary systems can more easily engage in penetration pricing than can open systems, whose many sponsors must somehow agree who will bear what portion of the early period sacrifice in profits, and who are unable to recoup the losses through higher prices in face of the onslaught of new entrants. If markets are biased against open systems, that bias should not be exacerbated with a version of software copyright principles that makes the evolution of open systems out of closed ones more difficult.

Finally, reverse engineering is necessary to make operational any policy that copyright should not extend to *de facto* standards. Otherwise firms "owning" the *de facto* standard will tend to release insufficient information to provide compatibility. For example, Microsoft regularly conceals interface standards as revealed in the recent Stac litigation.[28] Absent an ability to judge for themselves (via reverse engineering) whether the release of interface information was adequate, competitors would be at the mercy of the standard owner, whose incentive is often not to release (at least not in a full or a timely fashion) interface information that others need to compete with him.[29]

IV. APPLICATION TO SOFTWARE COPYRIGHT CASE LAW

In this section, we apply our economic analysis of software copyright to several recent cases. From an economic perspective, the Altai case seems correctly decided and even uncontroversial. In contrast, the logic of the Paperback court in finding liability appears at odds with our analysis. Fortunately, the recent Borland decision restored valid economic logic to disputes involving alleged infringement by a direct competitor in markets where user investments and network externalities are important.

The Sega and Nintendo cases, in our view, involve quite different issues that have not been recognized. While network externalities may be somewhat relevant in those cases, we suspect that the primary reason Sega and Nintendo attempted to

---

[25] Of course, competitors should not be allowed to sell copies of the expression (code) of the initial spreadsheet standard. Competitors would not have to do so to compete "on a level playing field" with the incumbent, since by assumption *ex ante* there are equally acceptable ways of expressing the idea.

[26] By similar argument, a software copyright holder should not be allowed to prevent competitors from utilizing syntax and command structures that users have learned from the initial spreadsheet. It would, however, be inefficient, and therefore should be illegal under trademark or copyright laws, to allow copying of screen displays. Copying of displays is not necessary to achieve compatibility, nor to allow users to continue employing the commands they have mastered, but it may create confusion as to who developed the software (and is responsible for supporting it).

[27] See Katz and Shapiro (1992) and Farrell (1989).

[28] In that litigation, Microsoft argued that undocumented calls in Windows were its own trade secrets. This position was inconsistent with its prior announcements to competing developers of Windows applications that Microsoft's in-house applications operations were not favored *vis-a-vis* independent software houses.

[29] Menell (1987) proposed compulsory licensing of software programs that have emerged as industry standards. This proposal probably could not substitute for the right to reverse engineer to achieve compatibility, given the standard-holder's incentive for less than complete revelation of relevant information.

restrict the sale of competing, compatible games was not to deny network economies of scale to potential competitors. Rather, their most likely goal was price discrimination. We argue that use of copyright to maintain price discrimination is inappropriate and probably unwise. Thus, the Sega and Nintendo courts arrived at the correct conclusion, albeit perhaps for the wrong reasons. Unlike many commentators, however, we do not see the case for allowing disassembly for complementary products as more compelling than the case for allowing the practice in developing directly competing products. Indeed, the case for allowing reverse engineering to develop directly competing products in markets where network effects are important is even stronger than the case for allowing reverse engineering for complementary products.

A. Recent Case Law Evolution Involving Direct Competition: Computer Associates v. Altai and the Lotus Cases

Several recent cases have addressed the extent to which owners of already established software products can invoke copyright infringement against new direct (*i.e.*, horizontal) competitors. There is no disagreement that exact copying of object code is infringement. These cases have all involved the extent to which new competitors can use a logical structure or user interfaces that are identical or similar to those of the incumbent.

The reasoning underlying the Altai decision is sound.[30] Altai, however, did not raise the issues of user investment or network externalities. Network effects were apparently not important in the market, so the distinction between *ex ante* and *ex post* merger of idea and expression did not arise. The court decided that, given the tight constraints imposed by the operating environment and the utilitarian task to be accomplished, there was an *ex ante* merger of idea and expression.[31] Earlier we defended this established principle of copyright law on economic grounds. If there is only one way to express an idea, then control over that expression grants control over that idea, significantly limiting the alternatives available to others. Given this control, the copyright holder could easily appropriate the value of something he did not create.

Several critics agreed with Computer Associates in its belief that there was no *ex ante* merger of idea and expression. They reasoned that Altai was free to develop and market its product in a non-IBM environment, to write separate applications for each IBM operating system, or to develop from scratch its own operating system for IBM mainframe computers. In practice, however, these alternatives would block access by Altai to a large block of customers. Not only would this harm those customers, but the high fixed costs involved in software development imply that denial of access to a significant share of the market would increase the average cost of supplying the remaining customers, perhaps so much that the remaining customers could not be served economically, and entry would be effectively blocked.

Two important cases involving Lotus have addressed whether the commands and the logic of the command structure in a software package should receive copyright protection. In Lotus v. Paperback, the defendant sold a spreadsheet program with a user interface that was very similar to the Lotus interface. The court found that although several individual elements of the interface could not be copyrighted because they were inherent (*i.e.*, merged) with the idea of the spreadsheet, the command structure of the interface was protectable

[30] Computer Associates International, Inc. v. Altai, Inc., United States Court of Appeals for the Second Circuit, January 9, 1992.

[31] Computer Associates and, later, Altai, sold job scheduling programs that operated within IBM mainframe environments and which contained program modules that allowed the same applications software to perform on different operating systems in that environment. The alleged infringement involved the converter or adaption routines that allowed the applications software to run on the different operating systems.

because numerous other ways to set up the spreadsheet's user interface were feasible. The defendant argued that, given Lotus' dominant position as the spreadsheet standard, customers would not regard interfaces that required user retraining as reasonable alternatives. The court rejected the argument, saying that it turned copyright principles on their head, and that Lotus should not be penalized for its success.

It would appear that the court implicitly assumed that *1-2-3*'s success did not increase the real cost to Paperback of expressing the unprotected idea of spreadsheets, *i.e.*, that *1-2-3* became the standard solely or primarily because of its own efforts. We would agree that, so long as the presence of a copyright-protected *1-2-3* reduced the price that consumers were willing to pay for Paperback's spreadsheet simply because *1-2-3* offered a superior product,[32] Paperback would have no case. But if Lotus imposed a real cost on Paperback by preventing potential customers of Paperback from taking advantage of network economies of scale (benefiting users of both *1-2-3* and the Paperback spreadsheet), then Lotus's attempt to reduce compatibility through its exercise of copyright protection harmed both competitors and consumers.

In those circumstances copyright protection that limited rivals' ability to achieve *1-2-3* compatibility would raise the cost they would face in developing and marketing competing spreadsheets, even those with significant new and enhanced features.[33] Unless the court was certain that no network externality existed (did it even consider the possibility?), copyright protection of this sort is unjustified (though patent protection might be justified[34]).

Our problem with Judge Keeton's logic in Paperback can also be seen by noting that he analyzed liability at the wrong time. Consistency requires that liability and harm be assessed at the same time.[35] Judge Keeton argued that the

[32] In economics jargon, in these circumstances the presence of a very successful *1-2-3* would impose a negative "pecuniary" externality on the makers of competing software. Economists generally would recognize that no market failure is indicated, and no correction is called for, if Lotus has gained by making its product more attractive (lower prices, higher quality) to consumers, even though this may reduce the amount that those consumers would be willing to pay for rival products. Imposing pecuniary externalities on one's rivals is just part of the normal process of competition.

[33] Using economics jargon again, such copyright protection would impose a "real" or "technological" externality on rivals. A real negative externality (pollution is the standard example) imposes net costs on the rest of society, as would be the case to the extent that Lotus has gained, not by making its product more attractive to consumers, but instead by making its rival's products less attractive to consumers. Real externalities are a source of market failure, and correction is called for unless the social costs of the correction exceed the costs of the externality. Here there are no social costs of correcting for the externality. Indeed, the social costs are actually negative, since correcting the externality simply involves not allowing companies to use the socially-provided justice system to enforce copyright principles that could impose substantial real externalities.

[34] Lotus claimed that a substantial fraction of its development costs for *1-2-3* were incurred in designing the user interface, and that its research made significant advances in the state of the art for user interfaces. If true, Lotus may have been entitled to patent protection. For an insightful discussion of alternative legal regimes that could be desirable to induce the proper level of investment in software if existing legal regimes are inadequate, see Samuelson, et. al. (1994) and Samuelson (1995).

[35] See Fisher and Romaine (1990). This influential article poses the question how much damages the owner of a high school yearbook containing Janis Joplin's autograph is entitled to if the yearbook is stolen. The answer depends on when the yearbook was stolen. If it was stolen before Janis Joplin became famous, the market value of the yearbook was small. It the yearbook was stolen after Janis Joplin became famous, the market value of the yearbook was high. Fisher & Romaine argue persuasively that it is very inefficient to allow plaintiffs to

copyright survived challenge under the merger doctrine based on the fact that alternative ways existed for Paperback to express the ideas in its spreadsheet. Therefore, Paperback was guilty of infringement. However, Lotus' argument that it had been damaged was based not on the *ex ante* situation, before users had invested heavily in learning the Lotus spreadsheet, but on the *ex post* situation where, precisely because of the heavy user investment, compatibility with Lotus was very important for competing spreadsheets.[36] *Ex post* there had been a significant merger of expression and idea, and this merger should have been recognized by the court in judging the extent to which Paperback had available realistic alternatives in each of the areas where Lotus claimed Paperback's spreadsheet was too similar to its own.

The Lotus v. Borland litigation addressed the proper extent of copyright protection over "proprietary" commands and command structures. Borland included the capability to display, execute and edit Lotus *1-2-3* macros in its Quattro Pro spreadsheet. Lotus claimed that Borland infringed its copyright over the menu command hierarchy which was copied into the Quattro Pro program. Borland responded by arguing that the *1-2-3* menu structure is a "system, method of operation, process or procedure" for controlling a computer program, and hence, is uncopyrightable under Section 102(b) of the Copyright Act.

Once again, Judge Keeton heard Lotus' complaint.[37] He ruled that *1-2-3*'s menus and submenus arranged in a specific order constituted a protectable expression of the underlying idea of controlling the spreadsheet program. He arrived at this conclusion by separating the abstract notion of a method of operation (*i.e.*, software control of a spreadsheet program) from its copyrightable implementation (*i.e.*, *1-2-3*'s menu structure).

In its review of the district court's decision, the appeals court concluded that such a separation was unwarranted.[38] Instead, it held that the menu structure was a program interface that was physically separable from the spreadsheet program but essential to its operation. This reasoning led it to agree with Borland that the *1-2-3* menu structure was a "method of operation," and so was unprotectable.

It is instructive that, while neither decision paid attention to the presence of network externalities, the appeals court did recognize the relevance and significance of sunk investments by users in learning the interface and in writing specialized complementary software. Customers spend considerable sums (by some estimates seven times Lotus' investment in *1-2-3*) to create customized macro programs, and would have to incur significant switching costs if those programs had to be rewritten for another spreadsheet program. The Court noted that:

> "Under the district court's holding, if the user wrote a macro to shorten the time needed to perform a certain operation in Lotus 1-2-3, the user would be unable to use that macro to shorten the time needed to perform that same operation in another program. Rather, the user would have to

---

sue for windfall damages, as would be the case if the yearbook was stolen before Joplin became famous, but damages were claimed on the value of the yearbook after her fame. The plaintiff would then be made more than whole, since the loss at the time of the crime was far smaller. Moreover, a rule allowing claims for windfall damages creates perverse incentives for litigation. No efficiency purpose is served by the litigation unless the expected harm from the wrong at least exceeds the plaintiff's costs of litigation at the time the wrong occurred.

[36] See Gandal (1994) for empirical evidence on the importance of Lotus compatibility in the spreadsheet business.

[37] Lotus Development Corp. v. Borland International, Inc. 831 F. Supp. 223 (D. Mass 1993).

[38] Lotus Development Corp. v. Borland International, Inc., U.S. Court of Appeals for the First Circuit, March 1993.

> rewrite his or her macro using that other program's menu command hierarchy. This is despite the fact that the macro is clearly the user's own work product."[39]

Judge Boudin's concurring opinion, in particular, also pointed out that the *1-2-3* interface had become the *de facto* standard largely or even primarily because of continued investments over time by users:

> "A new menu may be a creative work, but over time its importance may come to reside more in the investment that has been made by users in learning the menu and in building their own mini-programs—macros—in reliance upon the menu."[40]

Judge Boudin went on to question why Lotus should be allowed to use the advantage it gained from 1-2-3's status as the *de facto* standard to seize users' sunk investments:

> "So long as Lotus is the superior spreadsheet—either in quality or in price—there may be nothing wrong with this advantage. But if a better spreadsheet comes along, it is hard to see why customers who have learned the Lotus menu and devised macros for it should remain captives of Lotus because of an investment in learning made by the users and not by Lotus."[41]

Critics of this argument may contend, however, that even if users appear to have undertaken substantial sunk investments in learning the program and in writing complementary software, some or all of these costs may have been born by the developer. This might occur even if the developer did not directly compensate users for such investments. As noted above in our discussion of penetration pricing, a *de facto* standard's original developer may find it profitable to "internalize" the incremental benefits of the many sponsors by setting prices to initial adopters that are lower than those that would have maximized its profits in each period. Indeed, if users expect that the initial developer will eventually appropriate the value of those investments through its control over a *de facto* standard, users would rationally require compensation *ex ante* in the form of lower—perhaps even negative—prices for the initial software. With perfect and costless information and foresight, users could thus prevent the developer of a *de facto* standard from opportunistically seizing the value of their investment.

Given that information available to users is neither costless nor perfect, however, the potential for users to force developers to fully compensate users for their sunk investments in complementary products through penetration pricing may be too limited to rely on much in practice. In addition, as discussed above, for products with many users with differing willingness to pay (*i.e.*, different reservation prices for the product) setting prices that are initially very high and then decline (*i.e.*, price skimming) can be the most profitable strategy for the developer. Moreover, in such an imperfect world "new" users (those who have not yet chosen a spreadsheet program) as well as "old" users may be harmed if entrants cannot compete on an equal basis for sales to locked-in users. Given the high fixed costs involved in software development, denial of access to a significant share of the market will increase the average cost to an entrant of supplying just the new customers, allowing the incumbent to raise prices to new customers as well as to old.

Thus, while the presence of sunk user investments and scale economies may not be sufficient to ensure that extending copyright protection for software to cover *de facto* standards or interface specifications would be anticompetitive in every instance, the likelihood of such an outcome is sufficiently high that the automatic and uncritical

---

[39] Id. at 28-29.

[40] Id. at 33.

[41] Id. at 37.

protection provided by copyright is inappropriate when significant sunk investments by users are present.

The presence of sunk user investments is not, however, the only valid reason for denying copyright protection to interface specifications and *de facto* standards. As discussed above, the case is made even stronger when network externalities are present. In that event, broad copyright protection can allow the developer to appropriate the value of network externalities enjoyed by current customers. If, in addition, copyright protection raises the cost to prior users of switching to a competing product, and allows the first mover to block access to a sufficiently large number of customers that an entrant would be unable to cover its fixed costs, then such protection also grants to the first mover *ex ante* monopoly power over new users in the future.

Furthermore, the larger the first-mover's installed base of users, the greater compensation both new and old customers will demand to switch to the new product whether or not they have sunk investments in the current interface specification or *de facto* standard. Where the installed base of users is large, coordinating a simultaneous switching of a large share of those users over to a new *de facto* standard may in practice be an impossible task for an entrant. This implies that the Appeals Court's rejection of copyright protection when it allows appropriation of user sunk investments should be extended to rejection of copyright protection when network externalities are present, even absent any sunk investments by users.

Interestingly enough, in the case of Lotus *1-2-3*, both user investments and network externalities appear to be significant and to have been sufficient to transform a situation of many *ex ante* equivalent means of expression into a single means of expression that is *ex post* uniquely superior to all others. This suggests that an appropriate and consistent economic basis for denying copyright protection to *1-2-3*'s menu structure can be found in the legal doctrine of merger. As we stated in an earlier version of this paper:[42]

> "We would interpret Borland's position as a statement that network externalities have transformed an *ex ante* large number of alternative expressions into an *ex post* merger of the idea with one expression, and that much of the sunk investment incurred to create that advantage for the Lotus product was in fact incurred by consumers. Hopefully the Borland court will recognize the importance of these arguments, and grant them greater deference than did the Paperback court. Aspects of *1-2-3* have become *de facto* standards for spreadsheets, and the question the court should attempt to answer is whether the alleged infringement involves a *de facto* standard. If so, copyright restrictions preventing others from conforming to the standard are inappropriate."

While the term "merger" never appears in the decision, the Court's statements carry at least the flavor of the merger doctrine.[43] While it is not "idea" and "expression" that is merged here, the cause of the merger is the same: given network externalities and users' sunk investments in the implementation, the cost of introducing an alternative implementation (*i.e.*, the switching cost to the users) is increased significantly, leaving only one economical implementation of the uncopyrightable material.[44]

---

[42] Warren-Boulton, Baseman, and Woroch (1995, pp. 23-24).

[43] "Expression is not copyrightable because it is part of Lotus 1-2-3-'s "method of operation." Id. at 23; "Thus the Lotus command terms are not equivalent to the labels on the VCR's buttons, but are instead equivalent to the buttons themselves." Id. at 27. Emphasis added.

[44] Similarly, the *scenes a faire* doctrine prevents the first mover from appropriating the value of investments made by others (*e.g.*, when subsequent authors continue the development or recognition of a stock character), or the value to new users (readers) created by its use by other

B. Recent Case Law Involving Complementary Products: The Nintendo and Sega Cases

The Nintendo and Sega cases established a "fair use" copyright exception for developing complementary products. Defendants developed competing games that could run on Sega's or Nintendo's game hardware, and succeeded in making their games compatible with the Sega or Nintendo hardware by using disassembly to crack the "lock-out" code in the software embedded in the game hardware. Both courts determined that such reverse engineering was lawful to the extent that other means of attaining compatibility (or learning the unprotected ideas in the embedded software) were not available. These decisions have been criticized on the grounds that Accolade and Atari were competitors of Sega and Nintendo at the game level: Accolade and Atari were not providing a entirely new use for the hardware, in which case the hardware manufacturers might have reacted differently.

If a company develops a new (or even just an improved) compatible and complementary product, it may be hard to see how the copyright holder can claim to have been harmed, since the demand for the initial product will increase because of this new product. But the economics of vertical control (as control over a complement is usually described) is quite complicated. There are a variety of reasons why an upstream company would seek to control a downstream market.

A common motivation, which we suspect is the primary explanation for the behavior of Sega and Nintendo, is price discrimination. Since the potential buyers of a game system are likely to vary considerably in terms of the maximum amount that they would be willing to pay for a game system, a game manufacturer would like to price discriminate, charging users who place a high value on the game system a higher price than users with lower valuations. A game manufacturer cannot achieve this result simply by charging different customers different prices for the hardware. The manufacturer can neither easily identify which customers would be willing to pay a high price and which a low price, nor can it prevent those receiving it at the low price from reselling it to those who would otherwise have paid the higher price.

The standard solution to this problem is to find a complementary good, the demand for which can serve as a measure or proxy for how much the customer values the system. The manufacturer then requires that the customer buy the complement from the manufacturer, and marks up the price of the complement. For example, an inventor of a new type of razor might reasonably expect that customers that use a large number of blades would be willing to pay more for the razor than customers who use very few blades. If the inventor can ensure that no one else can supply blades that are compatible with his razor, he can combine a low price for the razor with high profit margins on the blades. This will result in much higher overall profits by extracting higher net revenues per razor from high-intensity users without having to forego profitable sales to low-intensity users. Similarly, for game systems, a customer's demand for games would seem to be a reasonable indicator of his reservation price for the game system. If so, a game manufacturer that can exclude others from providing compatible games (or charge suppliers of compatible games a high license fee) will find it profitable to take profits at the game level rather than at the hardware level.

The critical problem, of course, is excluding others from supplying the complementary product, especially given the high margins set by the manufacturer for the games. When a manufacturer attempts to exclude others by contract with buyers, this is referred to as a tying arrangement, and often leads to dire antitrust consequences. Here, Sega appears to have attempted to use the copyright laws to exclude other suppliers of compatible games in order to allow it to price discriminate among users.

---

users (recognition of a stock character) which we would regard as a network externality.

While economists regard the welfare effects

of such price discrimination as generally indeterminate *a priori*, and would thus generally support a rule of reason approach to such actions, antitrust case law has taken a much harsher view.[45] Whatever the determination, however, it would be ironic indeed for a supplier to achieve through copyright something that would be illegal if achieved through agreement.[46]

There are three possible approaches to this complex issue in intellectual property, each corresponding to an established antitrust rule.

1) *Per se* illegality for decompilation and disassembly (or for unauthorized copying in general), which corresponds to a rule of *per se* legality for vertical control.

2) Some sort of "rule of reason" inquiry into the economic effects of banning or limiting disassembly, corresponding to a rule of reason for vertical control.

3) *Per se* legality for decompilation and disassembly, corresponding to a rule that vertical control must be achieved through other means.

Given the *a priori* uncertainty about the effects of copyright rules that allow control over complements, the first option could only be defended if it were clear that disassembly or decompilation has undesirable effects in almost all circumstances and that it is too costly to determine when the effects might be benign. But given that end-users often engage in these practices to make software more useful to them—hardly a practice against the interests either of the software's developer or society—this position would appear to be untenable.

Either a "rule of reason" or *per se* legality for decompilation and disassembly can be defended in principle, although a rule of reason could impose substantial uncertainties as well as legal and enforcement costs. Absent some estimate of the costs and benefits from a rule of reason approach, therefore, *per se* legality for decompilation and disassembly appears to be the most appropriate rule.[47]

V. CONCLUSION

We have argued that copyright protection should be provided only when the probability of a false positive (*i.e.*, granting an unwarranted monopoly) is *de minimis*. But, as we have seen, when network externalities or significant user investment are present, the control over substitutes made possible by copyright of interface specifications or *de facto* standards can significantly harm competition and reduce welfare. In addition, when copyright is used to control complements, the welfare effects of such vertical control are at best ambiguous. We conclude, therefore, that copyright should not be used to block compatibility with a rival's product, whether that product is a complement or a substitute, and that copyright protection should thus not be extended to interface specifications or to *de facto* standards. Fortunately, with some minor exception, recent case law has taken steps in this direction by helping to clarify legal principles that prevent producers from gaining

[45] In general, the welfare effects of vertical control are ambiguous. Some motives for vertical control generally result in its being beneficial (maintaining the quality of complements or preventing free-riding). Other motives have ambiguous effects (price discrimination), and still others result in bad or inefficient effects (raising rivals' costs or increasing entry barriers). Making the correct diagnosis is often very difficult.

[46] The antitrust cases involved contracts (a contractual tie) that compelled users to purchase the complements from the manufacture. Sega sought to realize a similar result through creating a technological tie. Indeed, Sega apparently had to incur extra cost to create the technological tie. See Warren-Boulton (1978) for a discussion of the issue generally, and Greenstein (1990) for a discussion of the issue in the context of software.

[47] Under this proposal, a firm could still attempt to exercise vertical control, but it would have to choose a more transparent method.

excessive protection under copyright.

As noted in our introduction, we do not regard the specific policy decisions evaluated in this paper—*e.g.*, copyright of interface specifications, or *de facto* standards and decompilation—as reflecting a battle between creative producers and later arrivals attempting to free-ride on their work. All the participants in this debate and in its associated litigation create value. The debate is between producers who came first and producers seeking to both build on and advance the past work of the others. For the latter group, setting appropriate

REFERENCES

Arrow, Kenneth. "Economic Welfare and the Allocation of Resources for Invention." in *The Rate and Direction of Economic Activity*. Princeton: Princeton University Press, 1962.

Arthur, Brian. "Competing Technologies, Increasing Returns, and Lock In by Historical Events." *Economics Journal*, March 1989.

Barzel, Yoram. "Optimal Timing of Innovation," *Review of Economics and Statistics*, 1968.

Cabral, Luis, Salant, David and Woroch, Glenn. "Monopoly Pricing with Network Externalities." Draft, University of California-Berkeley, 1994.

David, Paul. "Clio and the Economics of QWERTY," *American Economic Review*, May 1985.

Farrell, Joseph. "Standardization and Intellectual Property." *Jurimetrics Journal*, Fall 1989, pp. 35-50.

Farrell, Joseph and Saloner, Garth. "Standardization, Compatibility, and Innovation." *Rand Journal of Economics*, Spring 1985, 16 (1), pp. 70-83.

Farrell, Joseph and Saloner, Garth. "Converters, Compatibility, and the Control of Interfaces." *The Journal of Industrial Economics*, March 1992, XL (1), pp. 9-34.

Fisher, Franklin and Romaine, R. Craig, "Janis Joplin's Yearbook and the Theory of Damages", *Journal of Accounting, Auditing & Finance*, Winter 1990, 5 (1), pp. 145-157.

Gandal, N. "Hedonic Price Indexes and an Empirical Test for Network Externalities." *Rand Journal of Economics*, Spring 1994.

Gilbert, Richard. "Symposium on Compatibility: Incentives and Market Structure." *Journal of Industrial Economics*, March 1992, XL (1), pp. 1-8.

Greenstein, Shane. "Creating Economic Advantage by Setting Compatibility Standards: Can 'Physical Tie-Ins' Extend Monopoly Power?" *Economics of Innovation and New Technology*, 1990, 1, pp. 63-83.

Katz, Michael and Shapiro, Carl. "Network Externalities, Competition, and Compatibility." *The American Economic Review*, June 1985, 75 (3), pp. 424-440.

Katz, Michael and Shapiro, Carl. "Product Introduction with Network Externalities." *The Journal of Industrial Economics*, March 1992, XL (1), pp. 55-83.

Landes, William and Posner, Richard. "An Economic Analysis of Copyright Law." *The Journal of Legal Studies*, June 1989, XVIII, pp. 325-363.

Matutes, Carmen, and Regibeau, Pierre. "Mix and Match: Product Compatibility Without Network Externalities." *Rand Journal of Economics*. 19(2) Summer 1988, pp. 221-234.

Menell, Peter. "Tailoring Legal Protection for Computer Software." *Stanford Law Review*, July 1987, 39, pp. 1329-1373.

Menell, Peter. "An Analysis of the Scope of Copyright Protection for Application Programs." *Stanford Law Review*, May 1989, 41, pp.1045-1104.

Miller, Arthur. "Copyright Protection for Computer Programs, Databases, and Computer-Generated Works: Is Anything New Since CONTU?" *Harvard Law Review*, March 1993, 106 (5), pp. 977-1073.

Samuelson, Pamela, et. al. "A Manifesto

Concerning the Legal Protection of Computer Programs," *Columbia Law Review*, 94, 1994.

Samuelson, Pamela. "An Entirely New Regime Is Needed." *The Computer Lawyer*, 12:2, February 1995.

Warren-Boulton, Frederick, Baseman, Kenneth, and Woroch, Glenn. "Copyright Protection of Software Can Make Economic Sense." *The Computer Lawyer*, 12:2, February 1995.

Warren-Boulton, Frederick. *Vertical Control of Markets: Business and Labor Practices*, Ballinger Publishing Company, 1978.