Chicago-Kent Journal of Intellectual Property

Volume 6 | Issue 1 Article 5

9-1-2006

Beyond Fair Use: The Right to Contract Around Copyright Protection of Reverse Engineering in the Software Industry

David N. Pruitt

BEYOND FAIR USE: THE RIGHT TO CONTRACT AROUND COPYRIGHT PROTECTION OF REVERSE ENGINEERING IN THE SOFTWARE INDUSTRY

David N. Pruitt[*]

Introduction

The law should promote technological innovation in the computer software industry, but not at the cost of prohibiting private parties from entering into freely negotiated agreements for the proper handling of their intellectual property. This Note's position is that federal copyright law is not intended to prohibit private parties from freely negotiating the way privately owned intellectual property may be used in a licensing agreement, if the terms are not unconscionable and do not violate the antitrust laws. Specifically, the focus of this Note is on common software licensing terms that prohibit licensees from engaging in a process known as reverse engineering. To the extent that the innovation in the software industry is harmed by such restrictive provisions, a free-market approach that allows parties to freely enter into licensing agreements will rely on new software providers to inevitably enter the market and provide more favorable licensing terms if they can do so at a price that the market will bear.

Reverse engineering is an important process used by computer programmers to build new software that is compatible with existing software. Although the process involves making copies of copyright-protected computer software, it falls under the Copyright Act's "fair use" exception.[1] The recently enacted Digital Millennium Copyright Act, which prohibits circumventing technological measures designed to protect access to a copyright-protected work, provides an exception for the limited purpose of reverse engineering to achieve compatibility between software products.[2] Still, many software vendors choose to further protect their intellectual property by including license provisions that prohibit reverse engineering by the licensee.[3]

This Note does not suggest that the process of reverse engineering is not important to the industry or unworthy of protection, but merely suggests that preventing the enforcement of these privately negotiated terms is outside the scope of copyright law. In the course of providing background on the process of reverse engineering, Part I discusses the economic importance of

[*] David N. Pruitt is an associate with Gozdecki & Del Giudice, LLP in Chicago, IL where he practices in business counseling, transactions and litigation. He graduated with honors from the Chicago-Kent College of Law in 2005. Mr. Pruitt would like to thank Clark Hedger for his insightful comments during the writing of this article.

[1] *See, e.g., Sony Computer Entertainment, Inc. v. Connectix Corp.*, 203 F.3d 596, 602 (9th Cir. 2000) (holding that reverse engineering meets the statutory criteria for fair use set out in 17 U.S.C. § 107).
[2] 17 U.S.C. § 1201(f) (West 2005).
[3] *See, e.g.,* Apple Computer, Inc., *Sample QuickTime 7 Software License Agreement* <http://www.apple.com/legal/sla/quicktime7.html> (last accessed Oct. 19, 2005) (Paragraph 2 of the sample agreement, entitled "Permitted License Uses and Restrictions," provides in part: "This License allows you to install and use one copy of the Apple Software on a single computer at a time . . . Except as and only to the extent expressly permitted in this License or by applicable law, *you may not copy, decompile, reverse engineer, disassemble, modify*, or create derivative works of the Apple Software or any part thereof.") (emphasis added).

reverse engineering to the computer software industry. Part II provides the statutory and doctrinal background of copyright law that initially established computer software as a protected medium. While most courts have found that private parties are free to negotiate license terms, even where they go beyond the protection offered by the Copyright Act,[4] commentators argue that allowing the enforcement of such licensing provisions is preempted under the Copyright Act and the Supremacy Clause to the United States Constitution because it contravenes the purposes of federal copyright law.[5] Part III discusses and refutes the primary arguments that anti-reverse engineering provisions are preempted by federal law. Part IV endorses a freedom of contract policy with regard to intellectual property licensing and argues that decreasing contractual freedoms is not the way to promote technological innovation. Instead, by allowing the parties to freely negotiate licensing terms, the market will determine what is most beneficial to the industry. Finally, Part V discusses the unconscionability and antitrust concerns that will invalidate certain types of anti-reverse engineering provision abuses that the market is incapable of curing.

I. What is Reverse Engineering and why is it Important?

A. The Process

Reverse engineering is defined as "the process of developing a set of specifications for a complex hardware system by an orderly examination of the specimens of that system."[6] Unlike "forward engineering," which is the process of using abstractions and designs to physically build a system, reverse engineering uses the system that is already built to identify the components and their interrelationships and then create a copy or representation of that system for another purpose.[7] Put another way, reverse engineering is simply "going backwards through the development cycle."[8] Furthermore, the process of reverse engineering does nothing to affect the original design of the existing software; it is simply used to build a new and entirely separate system.[9]

An understanding of the practical implications of this procedure depends upon the understanding of a few simple concepts in computer programming. When a programmer designs a piece of software, it is written in a source code language, for example, C++. The source code must then be translated into binary, or object code, which is an extensive combination of 1s and

[4] *See, e.g., Bowers v. Baystate Tech., Inc.*, 320 F.3d 1317 (Fed. Cir. 2003); *Davidson & Assoc. v. Jung*, 422 F.3d 630 (8th Cir. 2005).
[5] *See, e.g.,* Seungwoo Son, "Can Black Dot (Shrink Wrap) Licenses Override Federal Reverse Engineering Rights?: The Relationship Between Copyright, Contract, And Antitrust Laws," 6 Tul. J. Tech. & Intell. Prop. 63 (2004). *See also* Mark A. Lemley, Brief of Amici Curiae in Support of Petition for Panel Rehearing and Rehearing En Banc Of Defendant-Appellant Baystate Technologies, Inc., 2002 WL 32345615 (2002).
[6] Elliot J. Chikofsky and James H. Cross II, "Reverse Engineering and Design Recovery: A Taxonomy." IEEE Software, pp. 13-17, 13. IEEE Computer Society, Jan. 1990 (citing M.G. Rekoff Jr., "On Reverse Engineering," IEEE Trans. Systems, Man and Cybernetics, Mar.-Apr. 1985, pp. 244-252).
[7] *Id.* at 14-15.
[8] Wikipedia, the free encyclopedia <http://en.wikipedia.org/wiki/Reverse_engineering> (accessed Sept. 16, 2005) (citing R. Warden, "Re-engineering – a practical methodology with commercial applications," *Software Reuse and Reverse Engineering in Practice*, pp. 283 – 305, Chapman & Hall, London, England, 1992).
[9] *Id.*

0s, in order to be read by a computer.[10] The object code is imprinted on a silicon chip and commercially distributed, but the source code is usually not made available to commercial purchasers.[11] Some software, known as open source software, makes the source code available to its users.[12] Of course, since the purpose of reverse engineering is often to translate the software back into source code, distributing the source code along with the software eliminates the need to reverse engineer in most cases. Nevertheless, when one has possession of the commercially distributed object code only, there are a number of ways in which one can reverse engineer a new product. The first is to simply observe how the object code is read by the computer.[13] For example, a recent case concerned a group of reverse engineers who created software that allowed users to play games designed for Sony's popular Playstation video game console on their home computer, rather than on the game console.[14] One way the engineers sought to accomplish this goal was by observing Sony's basic input-output system ("BIOS") as it operated on a computer with software intended to simulate the actual Playstation hardware.[15] This way, the engineers were able to observe the signals going between the BIOS and other programs on the computer, allowing them to craft their own software with similar compatibility.[16]

The second method of reverse engineering that can be performed when someone has possession of the object code actually involves disassembling the software using a device called a "decompiler" to read the signals produced while the program is being run and then translating those signals back into the source code.[17] However, the decompiler does not provide the original programmers' annotations that provide vital instructions as to the functioning of the software.[18] Consequently, the engineer must copy the original code repeatedly and disassemble the software one instruction at a time in order to ascertain the functioning of each command.[19] As such, the process requires a great deal of skill and time.

B. Economic Consequences

The continued use or restriction of reverse engineering has serious economic implications for the computer software industry. The economic reason for engaging in reverse engineering is obvious: to make a compatible software product at a lower cost.[20] Reverse engineering is particularly crucial because achieving compatibility will allow a firm that makes applications for computer platforms, for example, Microsoft Windows or Sony Playstation, to compete without

[10] *Sega Enterprises, Ltd. v. Accolade, Inc.*, 977 F.2d 1510, 1515 n. 2 (9th Cir. 1992).
[11] *Id.*
[12] For a discussion of the history and implications of open source software, *see* Brian W. Carver, "Share and Share Alike: Understanding and Enforcing Open Source and Free Software Licenses," 20 Berkeley Tech. L. J. 443, 444-60 (2005).
[13] *Sony Computer Ent.*, 203 F.3d at 600 (citing Andrew Johnson-Laird, "Software Reverse Engineering in the Real World," 19 U. Dayton L. Rev. 843, 845-46 (1994)).
[14] *Id.*
[15] *Id.*
[16] *Id.*
[17] *Accolade*, 977 F.2d at 1515.
[18] *Sony Computer Ent.*, 203 F.3d at 600.
[19] *Id.*
[20] Paula Samuelson & Suzanne Scotchmer, "The Law & Economics of Reverse Engineering," 111 Yale L.J. 1575, 1580 (2002).

being required to develop its own computer platform.[21] A firm that makes a platform has to make a business decision as to whether it will allow other firms access to the information they need to make compatible products.[22] For example, Microsoft has chosen to exercise strict control over its application programming interfaces ("APIs"), which are necessary to make Windows-compatible applications, resulting in market dominance year after year.[23] In the late 1980's, Nintendo also kept a tight lid on its APIs in an attempt to corner the market on games that were compatible with the Nintendo Entertainment System.[24] However, if licensees are allowed to engage in reverse engineering, the market power of such a restrictive platform developer will naturally decrease.[25] While it might seem that this would be beneficial as it would increase competition, it would also decrease the incentives to invest in the development of computer platforms.[26] For example, in the video game industry, firms typically lose money on the consoles that they sell and hope to make it up by selling games and other accessories.[27] Thus, a platform developer in that industry could develop a superior console, but fail to realize a profit on its research and development expenditures because of a flooded game market resulting from reverse engineering.

Still, commentators argue that the process of reverse engineering is so difficult and time consuming that it would not eliminate the incentive for firms to engage in platform development.[28] The extremes are obvious enough. On one hand, reverse engineering is restricted through licensing practices and software is provided at a lower cost, but with less competition comes less innovation. On the other hand, reverse engineering is allowed to the full extent encouraging competition and innovation, but software comes at a higher cost because engaging in platform development is a higher risk in this scenario. Still to be seen is whether software developers that distribute open source software will be able to profit to the extent that there is an incentive to develop new and innovative products. These considerations are discussed at length in Section IV.

II. Copyright Protection of Computer Software

A. Generally

In *Sega Enterprises v. Accolade*, Sega sued Accolade under the Copyright Act for using reverse engineering to create video games that were compatible with the Sega Genesis game console.[29] Accolade argued that disassembling the object code of the Genesis console could not constitute copyright infringement because that would entail ascertaining the code's ideas and functional concepts that were not comprehensible to humans.[30] In essence, Accolade was arguing that the object code contained in computer programs was not entitled to protection under

[21] *See id.* at 1615-16.
[22] *Id.*
[23] *Id.* at 1619.
[24] *Id.* at 1617.
[25] *Id.* at 1621-22.
[26] *Id.*
[27] *Id.* at 1618-19.
[28] *Id.* at 1622.
[29] 977 F.2d at 1514.
[30] *Id.* at 1519.

the Copyright Act.[31] Ultimately, the court rejected the argument and held that computer software code can be protected.[32] Nonetheless, an overview of this issue is necessary in order to fully understand the more specific problem of whether reverse engineering may be prohibited in software licensing agreements.

Computer programs are protected under the Copyright Act of 1976 as "literary works."[33] The Copyright Act defines literary works as "works, other than audiovisual works, expressed in words, numbers, or other verbal or numerical symbols . . . such as books, periodicals, manuscripts, phonorecords, film, tapes, disks, or cards, in which they are embodied."[34] The legislative history provides further clarification that literary works do include "computer data bases, and computer programs to the extent that they incorporate authorship in the programmer's expression of original ideas, as distinguished from the ideas themselves."[35] Of course, the notion that the ideas themselves are not subject to copyright protection is not specific to computer software; rather, it applies to any medium that is protected under the Copyright Act.[36]

Copyrights protect the expression of ideas, but not the ideas themselves. To help distinguish between the two, courts developed the doctrines of "merger" and "scenes a faire."[37] On one hand, if a particular form of expression is necessary to convey an idea, or if there are only a small number of ways to convey a particular idea, then the idea and the expression "merge," resulting in the lack of copyright protection for the expression.[38] However, this doctrine has been difficult to apply in the context of computer software. For example, in one recent case an audit recovery service developed a software program that identified lost profits due to payment errors.[39] After the company's former employees started a competitor service and developed a similar computer program that contained almost identical user interface elements, it brought an action based on copyright infringement.[40] The court found against the plaintiffs because the similarities between the two pieces of software were merely processes to achieve the

[31] *Id.*
[32] *Id.*
[33] *Atari Games Corp. v. Nintendo of Am., Inc.*, 975 F.2d 832, 838 (Fed. Cir. 1992).
[34] 17 U.S.C. § 101 (West 2005).
[35] *Atari Games Corp.*, 975 F.2d at 838 (citing H.R. 1476, 94th Cong., 2d Sess. 54 (1976), *reprinted in* 1976 U.S.C.A.A.N. 5659, 5667).
[36] *See* 17 U.S.C. § 102(b) (West 2005) ("In no case does copyright protection extend to any idea, procedure, process, system, method of operation, concept, principle, or discovery, regardless of the form in which it is described, explained, illustrated, or embodied in such work."). *See also Computer Assoc. Int'l v. Altai, Inc.*, 982 F.2d 693, 703 (2d Cir. 1992) (Stating, "It is a fundamental principle of copyright law that a copyright does not protect an idea, but only the expression of an idea.").
[37] *Lexmark Int'l, Inc. v. Static Control Components, Inc.*, 387 F.2d 522, 535 (6th Cir. 2004).
[38] *Id.*
[39] *PRG Schultz Int'l, Inc. v. Kirix Corp.*, 2003 WL 22232771 at *2 (N.D. Ill. 2003).
[40] *Id.* at *3. Specifically, the similar user interface elements included "menus, toolbars, or right click menus for accessing common functions such as sorting records, inserting columns, hiding columns or coloring columns, dockable frames for organizing the user workspace or status bar for displaying table metrics, grid components for displaying data, tree controls for managing projects, dialogs or panels for creating or modifying formulas or for creating, modifying, or representing relationships between tables; and tabbed list controls for storing and displaying formulas." *Id.*

functionality of a recovery audit.[41] Consequently, it was impossible to separate the expression elements of the software code from the functional process elements.[42]

On the other hand, the doctrine of "scenes a faire" limits copyright protection to expression that does not necessarily follow as a matter of circumstance.[43] As applied to computer software, scenes a faire would not allow protection of software programming that is "dictated by practical realities," which might include hardware standard, compatibility requirements and standard industry practices.[44] Since there is high degree of overlap between these two doctrines, spending a lot of time deciding which doctrine applies is unnecessary. To that end, courts have focused on whether the idea at issue is capable of alternate modes of expression in a practical sense, as opposed to one that is merely theoretical.[45] For example, a telephone directory does not have to be organized alphabetically; rather, it could be organized by street address, birth date, age, or other individual characteristics.[46] However, the practical reality is that telephone directories are generally organized alphabetically and consumers have come to expect that to be the case.[47] Thus, that particular idea is not capable of an alternative form of expression under practical circumstances. Still, separating the expressive elements from the functional elements of a piece of computer code is difficult for someone who has little or no training in the technical aspects of computer programming, which includes most federal judges.[48]

Despite the inherent difficulties in determining which portions of software code should be given copyright protection, software as a whole has received generous protection by the federal circuit courts. Returning to *Accolade*, the court granted broad protection to the software at issue in that case and chose not to separate expressive elements from mere ideas.[49] Accolade argued that the object code in computer software could not be an expression of an idea because it was not something that a human could even comprehend.[50] However, the court took the approach that the Copyright Act does not distinguish between computer programs that can be read by the user and those that manage the computer system.[51] Moreover, the plain language of the Copyright Act even extends protection to works that can be understood "either directly *or with the aid of a machine*."[52]

When the Ninth Circuit decided *Accolade*, the Second Circuit was crafting its own analysis for determining the extent of copyright protection for computer software.[53] In *Computer*

[41] *Id.* at *4.
[42] *Id.*
[43] *Lexmark*, 387 F.2d at 535 ("Scenes a faire" literally means scenes that "must be done.").
[44] *Id.*
[45] *Id.* at 536.
[46] *Id.* (citing *Feist Pub'lns, Inc. v. Rural Tel. Serv. Co.*, 499 U.S. 340, 363 (1991) (holding that publisher of an alphabetically organized telephone directory did not meet the originality requirement of copyright protection.)).
[47] *Id.*
[48] *See* Mark A. Lemley, "Convergence in the Law of Software Copyright?," 10 High Tech. L.J. 1, 6-7 (1995) (commenting that although the separation of idea and expression is not a new concept in copyright law, the technical aspect of computer software make it a particularly difficult context for application).
[49] *Accolade*, 977 F.2d at 1519-20.
[50] *Id.*
[51] *Id.* at 1519 (citing *Apple Computer, Inc. v. Formula Int'l, Inc.*, 725 F.2d 521, 525 (9th Cir. 1984)).
[52] 17 U.S.C. § 102(a) (West 2005) (emphasis added).
[53] Son, *supra* n. 5, at 76.

Associates v. Altai, the Second Circuit formulated a judicial model for breaking down a computer software program into distinct sub-programs and then determined the appropriate level of protection for each.[54] The approach, called "abstraction-filtration-comparison" after its three steps, begins with the abstracting of the computer program at issue into distinct sets of instructions, or code, upon which higher levels of programming rely.[55] A computer program is essentially comprised of layers of commands in the form of code that all build off of each other.[56] The first step in this analysis is to deconstruct those layers.[57] Of course, trial courts will rely heavily on expert witnesses for this process. The second step, filtration, is an application of the copyright laws to each separate module that was abstracted in the first step.[58] Those modules that came as a result of efficiency or external factors and those that were merely ideas taken from the public domain can then be filtered out of the analysis because they are not entitled to copyright protection.[59] Finally, the two computer programs that are the subject of an infringement action can be compared based on the elements of the program that the court has determined are entitled to copyright protection.[60]

B. The Fair Use Defense

Since the process of reverse engineering involves making intermediate copies of computer software in an attempt to translate the object code into source code, one who engages in the process could be liable for copyright infringement even though the final product is not a copy.[61] Therefore, a reverse engineer has two arguments to rely on in order to avoid liability under the Copyright Act. The first is to argue that the computer code that was copied during the process was merely functional and not entitled to copyright protection. If the only portion of the code that was copied was that which allows compatibility, it is unlikely that there will be a case for infringement.[62] The second possible argument is that the copying was allowed because it was a fair use of the copyrighted material. To determine whether the use of a copyright is a "fair use," the Copyright Act established a four-factor balancing test.[63] The factors involve an examination of the purpose of the use, the nature of the copyrighted work, whether the work was copied in whole or in part, and the effect on the value of the copyrighted work.[64]

[54] 982 F.2d 693, 706 (2d Cir. 1992).
[55] *Id.* at 707.
[56] *See id.*
[57] *Id.*
[58] *Id.*
[59] *Id.* at 708-10.
[60] *Id.* at 710.
[61] *See Accolade*, 977 F.2d at 1518. The court recognized that the computer files generated from the disassembly program met the statutory definition of a "copy." *Id.*
[62] *See Lexmark*, 387 F.3d at 544 (finding that no infringement existed because the "lock-up" code contained in Lexmark's printer cartridge was not entitled to copyright protection, the court determined that it need not consider the fair use defense, but analyzed the issue nonetheless).
[63] 17 U.S.C. § 107 (West 2005) (providing that "In determining whether the use made of a particular work in any particular case is a fair use the factors to be considered shall include – (1) the purpose and character of the use, including whether such use is of a commercial nature or is for nonprofit educational purposes; (2) the nature of the copyrighted work; (3) the amount and substantiality of the portion used in relation to the copyrighted work as a whole; and (4) the effect of the use on the potential market for or value of the copyrighted work.").
[64] *Id.*

1. Purpose of the Use

Although using a copyrighted work for commercial purposes generally weighs against a fair use finding,[65] courts have looked the other way when it comes to products that result from reverse engineering.[66] The *Accolade* court found that the mere production of a competing product was not enough to negate fair use. [67] Instead, it found that the "purpose and character" of the copy was to study the functional requirements of the Sega Genesis console. [68] The creator's profitability motive was therefore an indirect result of using the copyrighted material. A more recent decision focused on the value of the copyrighted work itself.[69] In *Lexmark International v. Static Control*, Static Control used reverse engineering to create a microchip that would allow users of Lexmark printer cartridges to circumvent an authentication program that prevented certain discount printer cartridges from operating more than once.[70] Those who purchased Lexmark cartridges at a discount had already agreed to return the empty cartridges to Lexmark after one use, so the authentication program was merely a means of enforcing that agreement.[71] In evaluating a fair use argument, the court stressed that Static Control did not seek to exploit the value of the authentication program as it copyrighted work; rather, it used the program to permit printer functionality.[72] The court also found that even where profitability would normally weigh against a fair use finding, Static Control's ultimate goal of profitability was not based on the exploitation of Lexmark's copyrighted material.[73] Therefore, the use of a portion of the code that facilitates compatibility between programs is not considered a direct, commercial purpose and weighs in favor of a fair use defense.

2. Effect on the Value of the Copyrighted Work

Closely related to the purpose of the use is the effect of the use on the copyrighted work's value. The *Lexmark* court focused heavily on the value of the copyrighted work itself and not on the final product.[74] Although the value of the toner cartridges might have been affected, the court found that the market for Lexmark's Toner Loading Program was not.[75] Similarly, the mere fact that reverse engineering yields a product that competes against the copyrighted work does not negate fair use.[76] When Connectix used reverse engineering to create a similar BIOS to that of Sony's Playstation game console, it ultimately created a new system that would compete with the Playstation. [77] Still, the court did not consider the economic loss that resulted from competition to be the kind of loss in value of a copyrighted work that the Copyright Act was

[65] *Harper & Row Publishers, Inc. v. Nation Enters.*, 471 U.S. 539, 562 (1985).
[66] *See Accolade*, 977 F.2d at 1522 (9th Cir. 1992) (stating that the presumption of unfairness in copying a copyrighted work for commercial purposes can be rebutted depending on the characteristics of the specific use).
[67] *Id.* at 1523.
[68] *Id.* at 1523-24.
[69] *See Lexmark*, 387 F.3d 522.
[70] *Id.* at 530-31.
[71] *Id.*
[72] *Id.* at 544.
[73] *Id.*
[74] *Id.* at 545.
[75] *Id.*
[76] *Sony Computer Ent.*, 203 F.3d at 607.
[77] *Id.*

concerned with in evaluating fair use.[78] Moreover, relying on *Accolade*, the court distinguished between a copyright use that results in a competitive product and one that actually supplants the original product. [79] It found that the former could be a fair use while the latter was simply an infringement.[80]

3. Nature of the Copyrighted Work

Just as the First Amendment differentiates between more and less important forms of speech, the fair use analysis differentiates between more and less important copyrighted expression. As if it were announcing the difference between political speech and commercial speech, the Supreme Court recognized that "some works are closer to the core of intended copyright protection than others."[81] Much like commercial speech, computer software code falls outside the realm of typically protected works because it contains certain functional elements that are not protected under the Copyright Act.[82] However, as explained above, the reverse engineer must also copy the expressive elements contained in the software through the decompilation process in order to ascertain the functional elements of the computer software needed to produce a compatible product.[83] In *Accolade*, Sega argued that Accolade's copying of the expressive elements of its video game software, as opposed to simply copying the functional elements, prevented a fair use finding.[84] Nonetheless, the Ninth Circuit rejected Sega's argument because the unprotected, functional elements of the software could not be examined without copying the expressive elements.[85] Therefore, the video game program was given "a lower degree of protection than more traditional literary works."[86] Consequently, the nature of computer software also weighs in favor of a fair use defense with regard to reverse engineering.

4. Amount and Substantiality of the Portion Used In Relation to the Entire Work

When applied to reverse engineering, the "amount and substantiality" factor will almost always weigh against the reverse engineer, but will not prevent a fair use finding. Since reverse engineering's purpose is to work backward from an existing computer program to translate the unreadable object code into readable source code, the engineer must copy the entire program. However, as the Ninth Circuit found in both *Accolade* and *Sony*, the reverse engineer's ultimate use of the entire program is extremely limited; accordingly, this factor is not substantial in the reverse engineering context.[87]

C. The Digital Millennium Copyright Act

[78] *Id.*
[79] *Id.*
[80] *Id.*
[81] *Campbell v. Acuff-Rose Music, Inc.*, 510 U.S. 569, 586 (1994).
[82] *Sony Computer Ent.*, 203 F.3d at 603.
[83] *Accolade*, 977 F.2d at 1525-26.
[84] *Id.* at 1526.
[85] *Id.*
[86] *Id.*
[87] *See Accolade*, 977 F.2d at 1526, *see also Sony Computer Ent.*, 203 F.3d at 605-06.

Enacted in 1998, the Digital Millennium Copyright Act ("DMCA") strengthened the protection of copyrights by prohibiting the circumvention of technological measures that are used to protect copyrighted works.[88] However, the DMCA included an exception to this prohibition for reverse engineering that was being used to achieve "interoperability" between two programs.[89] In one sense, the DMCA was codifying the federal courts' interpretation of the fair use doctrine as applied to reverse engineering. The DMCA defines interoperability as "the ability of computer programs to exchange information, and of such programs mutually to use the information that has been exchanged."[90] Indeed, the exception is a limited one and does not grant an absolute right of reverse engineering. Still, some have argued that the Act may even place additional restrictions on the process.[91] Nonetheless, the protections and exceptions afforded by the DMCA are still "rights against the world,"[92] and do not affect the ability of two private parties to enter into a contract. Therefore, even though the DMCA specifically allows reverse engineering in some circumstances, it should not affect the ability of private parties to enter into a contract that restricts the right to reverse engineer. This is further evidenced by the policies underlying passage of the DMCA. The problem Congress faced was that the technology used by copyright owners to protect digital works was being outrun by technology that would allow others to circumvent that protection.[93] Consequently, the Act was primarily intended to be a "burglar's tools" statute targeting those who were using circumvention techniques to illegally obtain copyrighted digital materials, as opposed to those who entered into licensing agreements to obtain the materials.[94]

D. The Uniform Computer Information Transactions Act (UCITA)

The Uniform Computer Information Transactions Act ("UCITA") is a proposed uniform state law that seeks to settle the law surrounding contracts and licenses in the information technology field.[95] Often criticized for disproportionately favoring software vendors, UCITA has only been adopted in Maryland and Virginia, both of which have a large high technology presence.[96] Nonetheless, the UCITA provisions concerning reverse engineering of computer software are an important part of the legal landscape surrounding the controversy over anti-reverse engineering clauses in software licensing contracts.

The project that became the UCITA was originally intended to be an additional article of the Uniform Commercial Code, article 2B, but its status changed to the freestanding uniform act

[88] 17 U.S.C. § 1201(a)(1)(A) (West 2005).
[89] 17 U.S.C. § 1201(f) (West 2005).
[90] 17 U.S.C. § 1201(f)(4) (West 2005).
[91] *See* Carla Meninsky, "Locked Out: The New Hazards of Reverse Engineering," 21 J. Marshall J. Computer & Info. L. 591, 611 (Summer 2003) (discussing the possible liability under the DMCA anti-trafficking provisions for a software engineer that includes circumvention technology in software that is later offered for sale).
[92] *See infra* n. 103.
[93] Thomas A. Mitchell, "Copyright, Congress & Constitutionality: How the Digital Millennium Copyright Act Goes Too Far," 79 Notre Dame L. Rev. 2115, 2116-17 (Oct. 2004).
[94] *See id.* at 2127 (citing Susan W. Brenner, "Complicit Publication: When Should the Dissemination of Ideas and Date be Criminalized?," 13 Alb J. Sci. & Tech. 273, 403 (2003)).
[95] *See* Samuelson, *supra* n. 20, at 1627.
[96] Frances E. Zollers, et al., "No More Soft Landings For Software: Liability For Defects In An Industry That Has Come Of Age," 21 Santa Clara Computer & High Tech. L.J. 745, n. 198 (May 2005).

that now exists when the American Law Institute withdrew amidst widespread criticism.[97] As its name suggests, the Act applies only to computer information transactions; therefore, it would not implicate an intellectual property license concerning property other than computer software.[98] While the UCITA does not expressly authorize anti-reverse engineering provisions in licensing agreements, it does so implicitly with its widespread authorization of mass-market license agreements which routinely contain such terms.[99] On the other hand, a specific provision stating that license terms contrary to federal law are unenforceable leaves open the possibility that anti-reverse engineering provisions could be preempted.[100] Still, the overarching theme embodied by the UCITA is that parties are free to negotiate their own terms in a licensing agreement.

III. The Preemption Argument

A. Express Preemption

Opponents of anti-reverse engineering clauses argue that the clauses are preempted by federal law. Of course, it is not the contract itself that would be preempted, it is the state's contract law that allows enforcement of the contract. These opponents rely on two separate grounds in support of their argument: express preemption and conflict preemption.[101] The basis for express preemption is found in §301(a) of the Copyright Act, which states:

> [A]ll legal or equitable rights that are equivalent to any of the exclusive rights within the general scope of copyright… are governed exclusively by this title. Thereafter, no person is entitled to any such right or equivalent right in any such work under the common law or statutes of any state.[102]

In other words, the Copyright Act expressly preempts any state law that attempts to provide parallel protection of rights protected by the federal law.[103] For example, a state cannot pass a statute that prohibits any and all copying of computer software because that concerns a right that is equivalent to those conferred by the Copyright Act.[104] However, the Federal Courts of Appeal have inferred an exception to the strict preemption stated by Congress in the Copyright Act. When an additional element is required by the state law cause of action, the state law claim is outside the scope of the copyright protection. Therefore, the claim is not preempted by the Copyright Act.[105] Moreover, recent decisions have held that the Copyright Act does not preempt an action for breach of a software license agreement because it contains elements in addition to those required for a copyright claim.[106]

[97] Pratik A. Shah, "The Uniform Computer Information Transactions Act," 15 Berkeley Tech. L.J. 85, 87-88 (2000).
[98] *Id.* at 88.
[99] *See* Samuelson, *supra* n. 20, at 1627-28.
[100] UCITA § 105(a) (West 2006).
[101] Jacob A. Gantz, "(Private) Order(ing) in the Court?: How the Circuit Courts Should Resolve the Current Conflict Over Reverse Engineering Clauses In Mass Market Licenses," 36 Rutgers L.J. 999, 1022 (Spring 2005).
[102] 17 U.S.C. §301(a) (West 2005).
[103] *Davidson & Assocs. v. Jung*, 422 F.3d 630, 638 (8th Cir. 2005).
[104] *Vault Corp. v. Quaid Software Ltd.*, 847 F.2d 255, 269 (5th Cir. 1988).
[105] *Data Gen. Corp. v. Grumman Sys. Support Corp.*, 36 F.3d 1147, 1164 (1st Cir. 1994).
[106] *See, e.g., Altera Corp. v. Clear Logic, Inc.*, 424 F.3d 1079, 1089 (9th Cir. 2005).

Using precisely that analysis, *ProCD v. Zeinenberg* held that license agreements governing the use of computer software were not preempted by the Copyright Act.[107] The "additional elements" required by the state law claim were those required to show a valid contract, including mutual assent and consideration.[108] *ProCD* concerned a software licensee that violated the terms of use contained in a software license agreement and argued that its "shrink-wrap" nature rendered it invalid.[109] However, the licensee's secondary argument was that the software licensing provisions were preempted by § 301 of the Copyright Act. In a particularly persuasive analysis, Judge Easterbrook concluded that contract rights are not equivalent to the rights conferred upon parties by the Copyright Act because copyrights create exclusive rights against the world, while contracts only affect their parties and do not create exclusive rights that bind strangers.[110] In Easterbrook's view, allowing the Copyright Act to preempt licensing provisions would create a slippery slope that would ultimately allow video store customers to refuse to return videos within the specified period on grounds that the rental agreement was preempted by the copyright protection afforded to the video itself.[111] Consequently, the court emphatically rejected the licensee's express preemption claim.

Since *ProCD*, the federal circuits have uniformly rejected the express preemption argument in the context of a software licensing agreement.[112] In light of these precedents, a plaintiff seeking to invalidate a license provision containing a reverse engineering prohibition will face overwhelming odds if she tries to do so on express preemption grounds because the suit will always contain the additional elements pertaining to a valid contract. Still, at least one commentator has argued that the rights created by a contractual provision prohibiting reverse engineering are equivalent to those protected by the Copyright Act, and are preempted as a result.[113] This argument relies on the legislative history of the Copyright Act, which states: "[a]s long as a work fits within one of the general subject matter categories of sections 102 and 103, the bill prevents the States from protecting it even if it fails to achieve Federal statutory copyright [protection]."[114] Since reverse engineering is a fair use and fails to achieve copyright protection, the argument reads this nugget of legislative history to authorize federal preemption of licensing provisions that prohibit it.[115] However, this statement of congressional intent refers to an *affirmative* act by a state to give further protection to works that are not protected by the Copyright Act. Interpreting such language to include the *passive* action of a state to *allow* private parties to dictate the use of their own private property would expand the scope of § 301's preemption formula to include claims in contract and tort that are far beyond what Congress

[107] *ProCD, Inc. v. Zeidenberg*, 86 F.3d 1447, 1453-55 (7th Cir. 1996).
[108] *Id.*
[109] *Id.* A shrink-wrap license agreement is an agreement that is contained beneath the shrink-wrap of a commercially available software program that is not revealed to the consumer until after he or she has returned home and opened the box. Courts have generally concluded that these contracts are enforceable on grounds that the contract is not formed until after the consumer has opened the box containing the licensing terms and decided not to return the product. *See id.* at 1452-53.
[110] *Id.* at 1454.
[111] *Id.*
[112] *See Bowers*, 320 F.3d at 1325.
[113] Son, *supra* n. 5, at 117.
[114] *Id.* (citing H.R. Rep. No. 94-1476, 94th Cong., 2d Sess. 129-131, *reprinted in* 5 U.S.C.A.A.N. 5747 (1976)).
[115] *See id.*

must have intended.[116] By allowing two private parties to enter into a contract that prohibits reverse engineering of a purchased piece of computer software, the state is not protecting additional works within the scope of the Copyright Act, but rather allowing two private parties to protect their own property. Consequently, the judicially created "additional elements" test is sufficient because it distinguishes between a state-created cause of action that affirmatively protects rights within the scope of the Copyright Act and one that has only a limited effect on copyrights by virtue of private party decisions.

B. Conflict Preemption

The second argument that reverse engineering provisions in software license agreements are preempted by federal copyright laws is based on the supremacy clause to the United States Constitution.[117] Even those who make the argument that the Copyright Act expressly preempts anti-reverse engineering clauses in licensing agreements acknowledge that conflict preemption is a much more viable theory.[118] Conflict preemption applies when it is impossible to comply with both federal and state law, or when state law creates an obstacle to the objectives that Congress envisioned when it created the federal law.[119] Critics of anti-reverse engineering clauses argue that such a provision contradicts the seminal purpose of Congress when it enacted the Copyright Act.[120]

A recent case in the Eighth Circuit concluded that Congress did not intend to strip private parties of their right to freely negotiate contractual terms by including reverse engineering as a fair use defense under the Copyright laws.[121] In that case, the software provider was a company called Blizzard that developed and sold computer games for use on personal computers.[122] To further the sales of its computer gaming software, Blizzard developed an online network called "Battle.net" that allowed purchasers of its software to play against other purchasers through the network.[123] In order for users to log on to Battle.net, the system had to authenticate their Blizzard software in order to ensure that they were using a properly authorized, i.e. not pirated, piece of software.[124] At the same time, those who did in fact purchase the gaming software had agreed to a shrink-wrap license agreement that prohibited reverse engineering of Blizzard's

[116] Without the additional elements test, express preemption would affect state created rights far beyond the enforcement of licensing agreements. For example, in *Shively v. Bozanich*, 31 Cal. 4th 1230, 80 P.3d 676 (2003), the California Supreme Court decided a statute of limitations question with regard to a defamation action. In that case, the plaintiff, Jill Shively, brought an action against the district attorney in the O.J. Simpson criminal proceedings for allegedly defamatory statements that were made and later published in a book. *Id.* at 1238, 80 P.3d at 680. Although copyright law was not an issue in the case, we can presume that the published work was within the scope of copyright protection. The same logic that would preempt enforcement of anti-reverse engineering provisions in licensing agreements would also preempt this defamation claim because it involved a copyrighted work. Surely, Congress did not intend to preempt such a broad range of state law claims when it enacted the Copyright Act.
[117] U.S. Const. art. VI, cl. 2.
[118] Son, *supra* n. 5, at 118.
[119] *Jung*, 422 F.3d at 638.
[120] Gantz, *supra* n. 97, at 1022-23.
[121] *Jung*, 422 F.3d at 640.
[122] *Id.* at 633.
[123] *Id.*
[124] *Id.*

software for any reason.[125] The defendants produced a similar network through reverse engineering that allowed gamers to compete against each other online with the Blizzard products. However, the newly created network did not require the authentication code; thus, users of unauthorized software could also use the new network.[126] Blizzard sued the defendants arguing that the reverse engineering was a breach of the license agreement.

The *Jung* court considered the defendant's argument that conflict preemption precluded state enforcement of the anti-reverse engineering licensing provisions.[127] Defendants relied primarily on *Vault v. Quaid Software*, which invalidated a Louisiana statute that specifically provided for the enforcement of anti-reverse engineering terms in a licensing contract under the doctrine of conflict preemption.[128] However, *Jung* made a clear distinction between a state statute that *affirmatively* authorizes the use of anti-reverse engineering clauses, and a situation where the state merely enforces an agreement between two parties.[129] In other words, the *Jung* court adopted the logic that previous courts had applied to express preemption cases to hold that a license agreement cannot stand as an obstacle to the Congressional intent of the Copyright Act.

While these precedents validate private party licensing agreements that grant additional intellectual property rights, they may create an additional obstacle for the further enactment of the UCITA. The decision in *Vault* makes it clear that an affirmative state authorization of reverse engineering may be a significant impediment to the full purposes and objectives of Congress in terms of the Copyright Act. The UCITA stops short of specifically authorizing anti-reverse engineering in license contracts, but its specific authorization of mass market licenses that often include these provisions will undoubtedly give preemption proponents more ammunition. Others argue that the inclusion of a reverse engineering exception in the DMCA strengthens the conflict preemption argument.[130] While this argument lends further support to the result in *Vault*, it does not go so far as to say that enforcing a private contract would impede the Congressional intentions behind the copyright laws.

IV. Policy In Favor Of Freedom of Contract

Ultimately, a freedom of contract policy should be adopted with regard to software licensing. Such a policy will allow the market to determine on its own which licensing provisions will be most beneficial to the software industry. The Supreme Court has long held that the freedom to contract is not an absolute right, and that federal and state legislatures are free to restrict the right so long as the restriction is reasonably related to a non-discriminatory legislative purpose.[131] Nevertheless, the policy in favor of allowing private parties to freely negotiate their own contractual terms is a longstanding American legal tradition.[132] Central to

[125] *Id.* at 634.
[126] *Id.* at 636.
[127] *Id.* at 638.
[128] 847 F.2d 255, 268-70 (5th Cir. 1988).
[129] 422 F.3d at 640.
[130] Gantz, *supra* n. 97, at 1023.
[131] *See, e.g., West Coast Hotel Co. v. Parrish*, 300 U.S. 379, 398 (1937).
[132] *See, e.g., U.S. v. Grace Evangelical Church of South Providence Ridge*, 132 F.2d 460, 462 (7th Cir. 1942) (stating the rationale that "[m]en [and women] shall have the utmost liberty of contracting and that their agreements, when entered into fairly and voluntarily, shall be held sacred and enforced by the courts").

this policy is that the contracts must be entered into voluntarily and fairly. When a software vendor has unequal bargaining power, two circumstances unrelated to copyright law provide purchasers with baseline protections where the general freedom of contract policy could fail.[133] The first is when the bargaining power so unfairly favors the software vendor that the terms are unconscionable. The second is when a software vendor violates the Sherman Act, which prohibits agreements in restraint of trade and certain unlawful conduct by monopolists. Outside of those situations, the freedom of contract policy should be observed and software licenses should be enforced according to the terms of the agreement even when the protection of the agreement goes beyond the scope of copyright law. To the extent that the ability to reverse engineer is vital to the progression of technology, the market will fill such a demand.[134]

A. Unconscionability Concerns

The unconscionability doctrine exists mainly to protect against contracts that are grossly one-sided, or present unfair surprises to consumers.[135] Although unconscionability may be difficult to show, the doctrine provides an important check on a software vendor's ability to impose unreasonable terms on its customers. When a contract contains an unconscionable term, the Uniform Commercial Code gives courts the power to invalidate the entire contract, or enforce the remainder minus the unconscionable term.[136] The Code defines unconscionable clauses as those that "are so one-sided as to be unconscionable under the circumstances existing at the time of the making of the contract."[137] Unconscionability exists in a procedural and substantive form.[138] While substantive unconscionability is concerned with the end result of a contractual term, procedural unconscionability targets the circumstances that were in place when the contract was formed, i.e., bargaining power, sophistication of the parties and knowledge of the terms contained in the contract.[139] In order to invalidate a contract, both forms must be shown on a sliding scale, i.e., the more procedurally unconscionable a contract, the less substantively unconscionable it needs to be.[140]

Without a bright line rule for determining unconscionability, the application of the doctrine depends on how a particular court views all of the circumstances that exist when a contract is formed. There are a number of ways in which software license agreements are formed. They can be formed in a traditional fashion where multiple drafts of a document are passed back and forth between the parties until it is satisfactory to both parties. More commonly,

[133] *See generally*, Son, *supra* n. 5 (discussing the application of unconscionability doctrine and antitrust law to anti-reverse engineering provisions in software contracts).
[134] Critics of this proposition also raise the argument that the market cannot respond to provisions in mass-market licenses because it would require the collective action of consumers who do not plan on engaging in reverse engineering and accept the terms on a take-it-or-leave-it basis. *See* Samuelson, *supra* n. 20, at 1629. In essence, the argument is that the market cannot respond to terms that are not negotiated in the first place. However, the target of this argument should be the validity of shrink wrap agreements in general, not the ability of private parties to agree to a particular term.
[135] For a discussion of the evolution of the unconscionability doctrine, *see Maxwell v. Fidelity Fin. Servs.*, 184 Ariz. 82, 907 P.2d 51 (1995).
[136] U.C.C. § 2-302(1) (West 2005).
[137] U.C.C. § 2-302, Official Comment Para. 1 (West 2005).
[138] *Davidson & Assoc., Inc. v. Internet Gateway, Inc.*, 334 F.Supp.2d 1164, 1179 (E.D. Mo. 2004).
[139] *Id.*
[140] *Comb v. Paypal, Inc.*, 218 F.Supp.2d 1165, 1172 (N.D. Cal. 2002).

however, software licenses are the product of shrink-wrap and click-wrap agreements where the user does not agree to the terms until he or she opens the packaging or begins to install the product.[141] Under California law, contracts of adhesion, defined as a "standardized contract, which, imposed and drafted by the party of superior bargaining strength, relegates to the subscribing party only the opportunity to accept or reject it," are procedurally unconscionable.[142] At least in circumstances where the subscribers to a shrink wrap agreement are not sophisticated, California courts have found that click wrap agreements are procedurally unconscionable.[143] In general, sophisticated software companies (and even less-than-sophisticated ones) understand the purpose and the terms of a licensing agreement. Therefore, a company that knowingly violates an anti-reverse engineering clause in its license agreement and then later claims that the terms were unconscionable will have difficulty finding a court sympathetic to its argument.[144]

Even where procedural unconscionability exists, the subscriber still must show that the terms are *substantively* unconscionable. In *Combs v. Paypal*, the court found that a click-wrap license agreement requiring Paypal users to submit to arbitration was substantively unconscionable because it was completely void of mutual remedies.[145] In addition, because the disputes in *Paypal* involved small amounts of money, the court found that customers were effectively prohibited from seeking additional relief in the courts.[146] However, this argument has been rejected when specifically applied to anti-reverse engineering clauses.[147] Nonetheless, with the right facts, a software user could argue that prohibiting reverse engineering in the license agreement effectively prevents the user from correcting defects in the software that could be corrected if the user had access to the program's source code. In that case, the licensee's biggest hurdle would be to convince the court that it was not sophisticated enough to contemplate this scenario prior to entering into the agreement.

B. Antitrust Concerns

1. Sherman Act §1

The antitrust laws provide another check on the ability of software vendors to impose unreasonable terms in a licensing agreement. The basic types of antitrust claims are created under the Sherman Act. Sherman Act §1 makes it illegal to form an agreement in restraint of trade.[148] As Justice Brandeis recognized, the very essence of a contract or an agreement is to

[141] *See ProCD*, 86 F.3d at 1453-55 (upholding the validity of shrink wrap agreements).
[142] *Paypal*, 218 F.Supp.2d at 1172 (citing *Armendariz v. Found. Health Psychcare Serv.*, 24 Cal. 4th 83, 113; 99 Cal. Rptr. 2d 745 (Cal. 2000)).
[143] *Paypal*, 218 F.Supp.2d at 1173 (holding that a click wrap agreement where the average transaction was $55.00 was an invalid contract of adhesion).
[144] In *Internet Gateway*, 334 F.Supp.2d at 1179, the court rejected an unconscionability argument on grounds that the subscribers to the licensing agreement were not "unwitting members of the general public," but were computer programmers familiar with the technical language used in the licensing agreement.
[145] 218 F.Supp.2d at 1173-74. The license agreement in *Paypal* required the subscriber to submit to arbitration as its sole remedy, while Paypal was allowed to unilaterally restrict accounts and withhold funds until the customer was later determined to be entitled to the funds in dispute. *Id.*
[146] *Id.* at 1175.
[147] *Internet Gateway*, 334 F.Supp.2d at 1180 (holding that anti-reverse engineering clause is not a harsh or oppressive term).
[148] 15 U.S.C. §1 (West 2005).

restrain trade.[149] Thus, the courts have interpreted the statute to include not only an agreement in restraint of trade, but one that is *competitively unreasonable*.[150] The courts have developed two categories for determining whether an agreement in restraint of trade is competitively unreasonable: per se illegality and the rule of reason. Per se illegality concerns practices like price fixing[151] and market dividing[152] that are so blatantly anti-competitive that simply showing that they occurred is enough to show a violation. By contrast, the rule of reason is a flexible model of analysis that weighs all the facts and circumstances including the type of restraint, its likely anti-competitive effects, pro-competitive justifications and the parties' market power.[153]

An anti-reverse engineering clause in a licensing agreement at least arguably restrains trade because it limits a software user's access to information needed to produce comparable or even compatible products.[154] In terms of reasonableness, such a clause would be viewed under the rule of reason analysis because it does not fit into a recognized per se category. However, there is an academic trend that favors Judge (later President and Chief Justice) Taft's view that viewing §1 violations should be based on whether the restraint is "naked" or "ancillary."[155] "Naked" restraints are those which have no purpose other than restraining competition.[156] "Ancillary" restraints are "restraints that arguably aid productive business transactions."[157] Under this model, naked restraints are per se illegal, while ancillary restraints that are reasonably related to furtherance of a productive business transaction are legal unless the restraint gives the defendant monopoly power.[158] One commentator has argued that anti-reverse engineering clauses that prohibit the copying of material otherwise allowed under the fair use doctrine are naked, per se illegal restrictions.[159] Moreover, the argument contends that the effects of these clauses are unreasonable because they go beyond the protection of copyright law and may stifle technological advancement because it limits a competitor's ability to develop a compatible product.[160] While it is true that the potential inability of a competitor to produce a compatible product could stifle technological advancement to some degree, this argument goes too far. Anti-reverse engineering clauses are not naked restrictions. In fact, they further productive business transactions by allowing software producers to protect their investment in research and development and prevent others from free-riding off of their success.

As an ancillary restriction, anti-reverse engineering clauses will be evaluated under the rule of reason, or, alternatively, based on whether they further a productive business transaction.

[149] "But the legality of an agreement or regulation cannot be determined by so simple a test, as whether it restrains competition. Every agreement concerning trade, every regulation of trade, restrains. To bind, to restrain, is of their very essence." *Chicago Board of Trade v. U.S.*, 246 U.S. 231, 238 (1918).
[150] *See, e.g.*, *Nat'l Society of Prof'l Engineers v. U.S.*, 435 U.S. 679, 690 (1978); *Standard Oil Co. v. U.S.*, 221 U.S. 1, 65 (1911).
[151] *See, e.g.*, *U.S. v. Socony-Vacuum Oil Co.*, 310 U.S. 150 (1940).
[152] *See, e.g.*, *U.S. v. Topco Assocs.*, 405 U.S. 596 (1972).
[153] *Cal. Dental Ass'n v. FTC*, 526 U.S. 756, 782 (1999) (Breyer, J., dissenting).
[154] Son, *supra* n. 5, at 128-29.
[155] Thomas C. Arthur, "Farewell to the Sea of Doubt: Jettisoning the Constitutional Sherman Act," 74 Cal. L. Rev. 263, 329-45 (1986).
[156] *Id.* at 271.
[157] *Id.*
[158] *Id.* at 334.
[159] Son, *supra* n. 5, at 131.
[160] *Id.*

On one hand, this approach will protect software producers whose sole motivation is to protect the heavy investment they have already made in developing a software product. On the other hand, it will guard against cases where the motivation behind the agreement is to prevent healthy competition, or to drive others from the market. The approach will also allow a court to easily invalidate such a clause when the vendor's market power is extremely high. Therefore, §1 will allow anti-reverse engineering provisions to exist in licensing agreements, but will provide an important check on the abuse of licensing agreements.

2. Sherman Act §2

Section 2 is not concerned with the activities of small businesses, but instead prohibits unilateral efforts by a firm to monopolize or attempt to monopolize a market.[161] For monopolization claims under §2, the plaintiff must show that the defendant has possession of monopoly power in a relevant market and has willfully maintained this power or used it in an exclusionary manner.[162] Since a § 2 claim requires the plaintiff to show that the defendant has monopoly power in a relevant market, the claim's success hinges on how broadly or narrowly the market can be defined.[163] The more narrowly the market is defined, the more likely that the defendant's market share will constitute "monopoly power."[164] In addition, the relevant market refers to both product markets and geographic markets.[165]

Anti-reverse engineering clauses imposed by a firm with monopoly power in a relevant market might constitute willful and exclusionary conduct. One commentator offers the following example: a company wants to enter the web browser market, but needs to make the browser compatible with the leading operating system.[166] The leading operating system is produced by a firm with monopoly power in the market for operating systems.[167] In addition, the monopolist also produces a web browser that is bundled with its operating system.[168] Finally, the licensing agreement that comes along with the operating system strictly prohibits reverse engineering, even for interoperability purposes.[169] In many cases, this would be an example of illegal use of monopoly power because the firm that has monopoly power in the operating system market is trying to leverage it to acquire or maintain a monopoly in the web browser market. This is accomplished via the anti-reverse engineering clause because the firm seeking to enter the market would otherwise be able to use the process to develop a new web browser compatible with the dominant operating system. And the firm that is imposing the restrictive agreement is clearly not using it to protect its own research and development for the operating system. After all, if the firm were competing against anyone else in the operating system market, it would *want* other firms to be able to make web browsers compatible with its operating system because it would make them more competitive in their own market. Thus, imposing an anti-reverse engineering clause in a licensing agreement under these facts would likely be a §2

[161] 15 U.S.C. §2.
[162] *U.S. v. Grinnell*, 384 U.S. 563, 570-71 (1966).
[163] *See U.S. v. Aluminum Co. of Am. (Alcoa)*, 148 F.2d 416 (2d Cir. 1945).
[164] *See id.*
[165] *Brown Shoe Co. v. U.S.*, 370 U.S. 294, 324 (1962).
[166] Son, *supra* n. 5, at 142.
[167] *Id.*
[168] *Id.*
[169] *Id.*

violation. As a result, even under a freedom of contract policy with regard to software licensing, the antitrust laws will provide a level of protection to software purchasers preventing vendors from abusing the policy.

C. A Free Market Approach

Finally, if anti-reverse engineering clauses in licensing agreements are truly detrimental to technological innovation, then the market will produce an alternative to the restrictive licensing practices that include anti-reverse engineering clauses. To some extent, the market has already done this with the emergence of open-source software producers.

From an economic perspective, the reason that copyright law exists is to allow holders of copyrights to enter into contracts for the dissemination of their work in a more efficient manner.[170] Without copyright law, the owner could not efficiently exclude non-owners from obtaining the information, which would eliminate the incentive to invest in the high cost of producing new work.[171] At the same time, allowing the holder of the copyright to exercise a complete monopoly would also be inefficient. The monopoly price would be far enough above the marginal cost of each copy that many potential buyers would be excluded.[172] Moreover, assuming that forms of expression entitled to copyright protection are all built at least in part upon previously copyrighted works, the monopoly price would increase the costs of producing new works due to royalty fees and transactional costs required to obtain the existing works.[173] Therefore, copyright law does not grant an absolute monopoly to holders. Instead, it has created exceptions like the fair use doctrine in an attempt to strike a balance between the copyright holder's right and the user's right to access information.[174]

In order to maintain the balance sought by copyright law, those who make the investment required to develop and distribute computer software must be able to further protect their property rights as they see fit through licensing agreements. Critics of this argument instead suggest that further restriction of the right to reverse engineer through licensing agreements will impede innovation and discourage developers from building off of the ideas of others.[175]

Even if the critics are correct in their assertions that access to source code through reverse engineering, or just free distribution, is vital to the continued technological innovation in the computer software industry, an obvious solution is being ignored: let the market decide which terms are better for the industry. If access to source code is in such high demand, the market will produce software vendors that are willing to distribute open source software, or allow reverse engineering for limited purposes in licensing agreements. Open source software is software that allows users to access its source code and to modify and redistribute the software.[176] The Open Source Movement began in the 1970s when software was rarely bought or sold outside the

[170] *See* Niva Elkin-Koren, "Copyright Policy and The Limits of Freedom of Contract," 12 Berkeley Tech. L. J. 93, 98-99 (1997).
[171] *Id.*
[172] *Id.* at 99.
[173] *Id.* at 100.
[174] *Id.* at 100-01.
[175] *See* Lemley, *supra* n. 5.
[176] Carver, *supra* n. 12, at 450.

context of a hardware transaction as a response to the first few software vendors that attempted to withhold source code from its users through confidentiality agreements.[177] In the late 1990s, the Open Source movement received a jolt with the development of an open source operating system knows as Linux designed to compete against major operating systems such as Microsoft Windows.[178] Linux and most other open source programs were distributed using a free General Public License ("GPL"). The GPL required free distribution of the source code, freedom to improve the program and make those improvements public, and required that subsequent distributions be made under the GPL.[179] In addition, the GPL does not permit a distributor to charge licensing fees.[180] Therefore, firms that develop and license open source software make money by providing customer service, user manuals and other complementary services for which fees can be charged.[181]

In a sense, the emergence of open source software is a market response to restrictive licensing practices by commercial software firms. Open source software allows users to do all of the things that critics of anti-reverse engineering clauses in software licenses argue are fundamental to the industry. Software is freely distributed under the GPL and users are free to copy the source code that is required to make new programs that are compatible with old ones. Of course, open source is still in its infancy and whether the incentives to produce software under this model will persist is still an open question. Nonetheless, the fate of open source software will say a lot about which licensing practices are most beneficial to the industry.

As the Open Source movement continues, one of the following two scenarios is likely to emerge, which will ultimately reveal which type of marketing practice is better for the industry. The first possible scenario is that the quality of open source products will become just as high as those that are commercially available and firms will be able to make a profit by selling complementary services. In this scenario, commercial software vendors will have to adhere to the new licensing practices in order to compete. Thus, one can assume that overly restrictive licenses that prohibit reverse engineering among other things were not necessary to protect the investment required to develop new products.

The second scenario is that open source products remain in the market but do not increase in market share and are not able to compete with the quality of commercially available products. If this were to happen, the most likely reason for it would be that firms are unable to recover their development costs through the complementary services that are offered in conjunction with the software. Of course, antitrust concerns could also contribute to the struggle to increase market share. However, since the law already provides a remedy for an antitrust violation, one must assume a lack of antitrust violations for the sake of this argument. The result of this scenario is that commercially available products will continue to dominate the market. Assuming the contracts are freely negotiated and free from antitrust violations, the inference

[177] *Id.* at 445.
[178] Robert W. Gomulkiewicz, "How Copyleft Uses License Rights To Succeed In The Open Source Software Revolution And The Implications For Article 2B," 36 Hous. L. Rev. 179, 184 (Spring 1999).
[179] David S. Evans & Anne Layne-Farrar, "Software Patents And Open Source: The Battle Over Intellectual Property Rights," 9 Va. J. L. & Tech. 10, 16 (2004).
[180] *Id.*
[181] *Id.* at 18.

must be that restrictive licensing practices are necessary to create incentives to develop new, high quality computer software products.

Ultimately, this market-based approach to determine which licensing practice is most beneficial to the industry is most in tune with the economic goals of copyright law. Since copyright law aspires to provide an efficient level of protection to a copyright owner in order to be sure that there is still an incentive to produce new works, it would be inefficient not to allow copyright holders to engage in licensing practices that do the same. Indeed, if the anti-reverse engineering protections currently contained in software licenses are not required to preserve this incentive, the market will make that decision.

Conclusion

The debate over the enforceability of anti-reverse engineering provisions will only increase as we proceed into the digital age. However, the market should not be regulated to the point that software vendors are unable to fully protect their investments in developing new intellectual property. While copyright law does not fully protect an owner from prohibiting licensees from engaging in reverse engineering, it also does not prevent that owner from demanding additional protections of his or her copyright in a private transaction. Nonetheless, critics argue that public policy in favor of innovation overrides the freedom to contract under these circumstances. Therefore, a state should not be allowed to enforce these private agreements that go further than the express protections of federal copyright law. Before we run to Congress demanding a federal law that expressly preempts state enforcement of anti-reverse engineering clauses, we must give the market a chance to decide which licensing practices are truly the most beneficial to the industry.