Harvard Journal of Law & Technology
Volume 31, Special Issue Spring 2018

THE NEW WAVE: COPYRIGHT AND SOFTWARE INTERFACES IN THE WAKE OF *ORACLE V. GOOGLE*

*Fred von Lohmann**

TABLE OF CONTENTS

I. INTRODUCTION

Imagine you are interested in starting a car company. Perhaps your cars will have innovative electric drivetrains, like Tesla's. Perhaps you will cater to a niche audience that is not served by mainstream automobiles, following the example of companies like smart and Morgan. Perhaps you have something new and entirely novel to introduce to the automotive market.

Now imagine that, as a condition of entering the market, you first must design a new interface for your car — something different from the steering wheel and pedals with which all of your potential customers are familiar. In order to try out your car, customers would first have to learn this new "method of operation" (to use the terminology of 17 U.S.C. § 102(b)). Perhaps you are up to the challenge — perhaps gestures or a touch screen? But this would be a formidable obstacle, as your potential customers would literally have to relearn how to drive in order to try your products. If that hurdle was not enough to put you off the idea of entering the market (entrepreneurs are nothing if not optimistic!), it might be enough to put off any potential investors. Certainly, the requirement to create a new interface would be a

* Legal Director, Copyright, Google LLC. The views herein represent the views of the author, not necessarily those of Google LLC.

substantial barrier to entry for your business and would represent a significant switching cost for your potential customers.

Fortunately, copyright law has many limiting doctrines that would make it impossible to assert copyright protection over the steering wheel and pedals of an automobile.[1] But as Professor Menell's article makes clear, while the stakes for software interfaces are just as high as for automotive interfaces, the copyright law answers are far less settled for software interfaces. The *Oracle v. Google* litigation — and the cases that have been brought in its wake — raises the very same questions posed by the automotive hypothetical: questions about switching costs, barriers to entry, and network effects. While these kinds of questions may, at first, seem far afield from copyright law, we can expect them to recur with increasing frequency in copyright cases in the years to come. After all, software will likely mediate more and more of the technologies we depend on. And interfaces are the steering wheels and pedals by which we operate software.

Those engaged in the copyright debate around interfaces fall roughly into three camps. First are those who agree with Professor Menell (I count myself among their number) that interfaces ought not be protectable by copyright, and who are worried that "[c]ompanies could use API strategies to lock in consumers and lock out competitors."[2] On this view, granting copyright protection to software interfaces (or "methods of operation" generally), permits the copyright owner not only to recoup its own investment, but also to unfairly and inefficiently capture the value of independent investments by its customers. The second group lies at the other end of the spectrum, untroubled by the "rise of the API copyright dead." Their reasoning is that the very purpose of copyright law is to create barriers to entry in the name of spurring investment and innovation. For them, once an interface has cleared copyright law's extremely low threshold of "creativity," there is nothing wrong with exclusive rights quelling free-riding by competitors. The third group takes a middle position, trusting in copyright's existing affirmative defenses, such as fair use and scènes à faire, to sort outcomes that enhance social welfare from outcomes that reduce competition to the net detriment of society.

---

1. Most importantly, copyright does not protect the design of useful articles, unless the article's design includes pictorial, graphical, or sculptural features that are independent of the utilitarian aspects of the article. *See* 17 U.S.C. § 101 (2012); Star Athletica LLC v. Varsity Brands, Inc., 137 S. Ct. 1002, 1008 (2017).

2. Peter S. Menell, *Rise of the API Copyright Dead?: An Updated Epitaph for Copyright Protection of Network and Functional Features of Computer Software*, 31 HARV. J.L. & TECH. (SPECIAL EDITION) 305 (2018); *see also* Pamela Samuelson, *Functionality and Expression in Computer Programs: Refining the Tests for Software Copyright Infringement*, 31 BERKELEY TECH. L.J. 1215, 1258–59 (2017) ("[f]reedom to reuse APIs, insofar as they are necessary for interoperability, promoted healthy competition and ongoing innovation in the software industry.").

Professor Menell does a thorough job of explaining why his view is the best of the three, based on statutory construction, jurisprudential evolution, and economic policy. Rather than restating those arguments, this comment aims to supplement Professor Menell's treatment by reviewing four software interface cases filed in the wake of the *Oracle v. Google* litigation. Not only are the "copyright API dead" rising after decades of relative quiet, they appear to be rising with increasing frequency. Taken together, they lend further support to Professor Menell's arguments and bear out the policy concerns that he sets out. These four cases also lend further credence to Professor Menell's view that affirmative defenses are an inadequate correction for overbroad software interface copyright protection.

II. THE NEW WAVE OF SOFTWARE INTERFACE CASES AFTER *ORACLE V. GOOGLE*

As Professor Menell explains, copyright cases involving software interfaces were rare in the decades between the *Lotus Dev. Corp. v. Borland Int'l, Inc.*[3] and *Oracle Am., Inc. v. Google Inc.*[4] litigations. In the wake of *Oracle v. Google*, however, there have been a spate of such cases, with their rate of appearance accelerating after the Federal Circuit's ruling approving the notion that interfaces can be protectable by copyright.

A. SAS v. WPL

The earliest in this "new wave" of interface cases is *SAS Inst., Inc. v. World Programming Ltd.*,[5] filed in the Eastern District of North Carolina in 2010. The facts are reminiscent of the early interface cases that Professor Menell discusses — a leading market incumbent sues a new market entrant for studying the interface of an existing software application, using that knowledge to reimplement those interfaces, and creating a competing product.

As with most real-world litigations, the case involves a number of procedural complexities (including a parallel litigation in the United Kingdom that made its way to the Court of Justice of the European

3. 49 F.3d 807 (1st Cir. 1995), *aff'd by an equally divided Court*, 516 U.S. 233 (1996).

4. 750 F.3d 1339 (Fed. Cir. 2014).

5. 64 F. Supp. 3d 755 (E.D.N.C. 2014), *aff'd in part and vacated in part*, 874 F.3d 370 (4th Cir. 2017). This case was filed a few months before *Oracle v. Google*, but was stayed for several years as parallel litigation took place in the UK. The appeal drew four dueling amicus briefs, all of which addressed the copyrightability of software interfaces and the Federal Circuit's ruling in *Oracle v. Google*. Nevertheless, the court of appeals found that the copyrightability issue need not have been decided because the plaintiff had achieved complete relief under its breach of contract claim. Accordingly, the court vacated as moot the district court ruling on the copyrightability issue. 874 F.3d at 389–90.

Union, Europe's highest court[6]) and non-copyright claims (in this case, breach of contract, on which SAS ultimately prevailed[7]). But the contours are simple enough. SAS is a software company based in North Carolina that makes the "SAS System," a suite of software applications used by enterprises to perform statistical data analysis.[8] The SAS System is the leading application in this market segment. In order to use the SAS System, its users must first create their own programs, written in the SAS programming language.[9] Accordingly, SAS customers face high switching costs if they want to try a different statistical analysis application, as they have to rewrite programs that they already wrote for the SAS System, in order to use those programs in conjunction with a different application.

The defendant, a UK company called World Programming Limited ("WPL"), identified an opportunity here. If it could independently develop an application that interoperated with the programs that SAS customers had already written, it would be able to compete directly with SAS. In order to create its competing application, WPL studied publicly available documentation about the SAS programming language, the behavior of a legitimately acquired "learning edition" of the SAS System, and the operation of the SAS System in the hands of an existing SAS customer.[10] This enabled WPL to create its own interoperable application called the World Programming System, designed to run the programs written by SAS customers — in the words of SAS, "a cheaper drop-in replacement."[11]

SAS responded by suing for copyright infringement. SAS argued that WPL had infringed its copyrights "by using certain software language functions and by copying the resulting output formats that are produced when a user runs those language functions through the SAS System."[12] In its briefing on appeal, SAS refers to these elements as the "input and output formats" of the SAS System.[13] With respect to the "input formats," SAS characterizes these as "a 'simple set of . . .

6. *SAS*, 64 F. Supp. 3d at 760–61; *see also* Jonathan Band, *The Global API Copyright Conflict*, 31 HARV. J.L. & TECH. (SPECIAL EDITION) 615 (2018) (describing the course and outcome of the European litigation).

7. *SAS*, 64 F. Supp. 3d at 769–74.

8. *Id.* at 759, 761.

9. *Id.* at 762.

10. *Id.* at 764–67. SAS alleged that WPL nevertheless breached the license agreement that governed the use of the "learning edition" of the software, but the resolution of that question was not material to the copyright claim.

11. Redacted Brief of Appellant/Cross-Appellee at 1, SAS Inst. Inc. v. World Programming Ltd., 874 F.3d 370 (4th Cir. 2017) (Nos. 16-1808, 16-1857) [hereinafter SAS Opening Brief].

12. *SAS*, 64 F. Supp. 3d at 775.

13. SAS Opening Brief, *supra* note 11, at 3–4 ("WPL meticulously copied the input and output formats of the SAS System, which reflect countless hours of creativity on the part of SAS, its statisticians and programmers.").

concise commands' to request a comprehensive analysis."[14] With respect to the "output formats," SAS describes these as "the tables, graphs, and other forms of output" that are produced by the SAS System from user programs.[15] In other words, the heart of the SAS copyright claim is focused on the interface used by SAS customers to operate the SAS System, as SAS itself admitted:

> SAS is unlike commonly used consumer software — such as a web browser, word processing program, or videogame — whose users interface with the software by clicking a mouse, moving a joystick, or typing text into a box. In contrast, SAS users interface by issuing written instructions to the software. Those instructions are provided by users as text files containing the required instructions and are generally referred to as "SAS Programs." SAS Programs are written in a high-level programming language developed and maintained by SAS and known as the "SAS Language." With a set of commands, a user can instruct the computer to access and arrange data and then perform a comprehensive analysis. Different from a computer programming language like FORTRAN or C (the underlying programming language used by SAS to write its own software), the SAS Language allows a user to cause the SAS System to process and analyze data with concise written instructions that would otherwise require "literally hundreds of thousands of lines of code" in a low-level programming language.[16]

The district court, however, was unpersuaded. It granted summary judgment to WPL, holding that all WPL had done here was copy the "SAS programming language," and languages are not protectable copyright subject matter:

> In essence, by asking the court to find that defendant's software infringes its copyright through its processing of elements [of] the SAS Language, plaintiff seeks to copyright the idea of a program which interprets and compiles the SAS Language — a language

14. *Id.* at 6.
15. *Id.* at 7.
16. *Id.* at 7–8 (citations omitted).

> anyone may use without a license. However, copyright law provides no protection to ideas.[17]

On appeal, SAS contended that these "input and output formats" should be copyrightable because its developers could have chosen a different structure of command names and output possibilities.[18] In other words, SAS asserted copyright in the creative choices made in naming the commands and defining the output possibilities that its customers must use in order to operate the SAS System.

This sounds reminiscent of Oracle's copyrightability theory. And, for that matter, it echoes Lotus' position regarding its menu commands in *Lotus v. Borland*.[19] In that case, which Professor Menell describes in more detail,[20] Lotus contended that the menu command hierarchy for its Lotus 1-2-3 spreadsheet was protectable by copyright. Lotus reasoned that it could have chosen different command names, and arranged them differently, and thus its competitor, Borland, should be prohibited from copying the names and arrangement.[21] The First Circuit flatly rejected that view: "The fact that Lotus developers could have designed the Lotus menu command hierarchy differently is immaterial to the question of whether it is a 'method of operation.'"[22]

In the end, the Fourth Circuit declined to resolve the question, vacating as moot the district court's copyrightability ruling in *SAS v. WPL*.[23] But for purposes of this discussion, the overall contours of the dispute remain relevant. WPL's brief on appeal puts the matter most directly:

> Extending copyright protection to the SAS language, as [SAS] proposes, would impermissibly expand the scope of [SAS's] copyright and grant a monopoly over the SAS language (and indeed over all SAS language programs written by users), prohibiting others from developing software that compiles/interprets the SAS language, which [SAS] does not own.[24]

---

17. *SAS*, 64 F. Supp. 3d at 776 (citing 17 U.S.C. § 102(b) (2012); Feist Publ'ns, Inc. v. Rural Tel. Serv. Co., 499 U.S. 340, 344–45 (1991)).

18. SAS Opening Brief, *supra* note 11, at 52–55.

19. *See* Lotus Dev. Corp. v. Borland Int'l, Inc., 49 F.3d 807, 816–18 (1st Cir. 1995).

20. Menell, *supra* note 2, at 336–340.

21. *Lotus Dev. Corp.*, 49 F.3d at 816–18.

22. *Id.* at 816.

23. SAS Inst. Inc. v. World Programming Ltd., 874 F.3d 370 (4th Cir. 2017).

24. SAS Opening Brief at 45–46.

In short, the story here is fundamentally one of switching costs for SAS customers, and whether copyright law should force those customers to rewrite all of their own code as a precondition of trying a competing statistical analysis system. By failing to resolve the copyrightability question, the Fourth Circuit missed a valuable opportunity to choose between the competing approaches taken by the First Circuit in *Lotus v. Borland* and the Federal Circuit in *Oracle v. Google*.

B. Synopsys v. ATopTech

The second in the "new wave" of interface cases is *Synopsys, Inc. v. ATopTech, Inc.*,[25] filed on June 26, 2013 in the Northern District of California. This dispute pitted an incumbent market leader in the field of electronic design automation ("EDA") software, Synopsys, against a smaller competitor, ATopTech. At issue were "place-and-route" software applications that customers use to plan the layout of a chip and the electrical connections among its various components.[26]

Synopsys' copyright infringement claim was premised on ATopTech's copying of input and output formats[27] — essentially the same theory pressed by SAS in its battle with WPL. With respect to input formats, Synopsys asserted infringement of "the combination of names and syntax comprising PrimeTime's and GoldTime's input formats."[28] Synopsys also claimed copyright over output formats, "such as including a line of asterisks to demarcate the initial, overview information for the report, using a string of dashes to provide separation before output data, and including flag and reason columns to label the output."[29] These interface elements constitute the method of operating Synposys' PrimeTime and GoldTime software; a user cannot operate software without knowing how to issue commands and what to expect in return. By supporting the same input and output formats used by Synopsys software, ATopTech promoted interoperability and reduced the switching costs that potential customers otherwise face. For this, it found itself embroiled in years of expensive litigation.

---

25. No. 13-CV-029652965 (N.D. Cal. filed Jun. 26, 2013) (ECF No. 1).

26. Synopsys, Inc. v. ATopTech, Inc., No. 13-CV-02965, 2016 WL 6158216, at *1 (N.D. Cal. Oct. 24, 2016).

27. *See* Synopsys, Inc. v. ATopTech, Inc., No. 13-CV-02965, 2016 WL 80549, at *1 (N.D. Cal. Jan. 7, 2016); Defendant ATopTech, Inc.'s Trial Brief in Support of Filtration, Synopsys, Inc. v. ATopTech, Inc., No. 13-CV-02965 (N.D. Cal. filed Mar. 7, 2016) (ECF No. 667).

28. Pl. Synopsys, Inc.'s Opp. to ATopTech's Mot. *in Lim.* No. 1 at 6, Synopsys, Inc. v. ATopTech, Inc., No. 13-CV-02965 (N.D. Cal. filed Feb. 2, 2016) (ECF No. 571). The precise details of the interface, as well as whether ATopTech copied all of the input and output formats or a subset, are obscured by the redactions made by the parties to protect their proprietary information.

29. *Id.* at *14.

Over the course of the litigation, the parties faced off across an array of claims and counterclaims, including patent infringement, breach of contract, and antitrust claims. In the end, however, after a three-week jury trial, the plaintiff prevailed solely on its copyright claim and was awarded $30.4 million.[30] ATopTech subsequently filed for bankruptcy protection, resulting in a consent judgment that precluded further appeal from the copyright verdict.[31]

C. Cisco v. Arista

The third in the "new wave" of interface cases after *Oracle v. Google* is *Cisco Systems Inc. v. Arista Networks, Inc.*,[32] filed in the Northern District of California in December 2014. The basic contours of the dispute once again echo those of previous interface cases. The plaintiff, an incumbent market leader, brings a copyright claim against a competitor who tries to mitigate customer switching costs by reimplementing the incumbent's software interface.

In this case, the plaintiff is Cisco, a market leader in Ethernet switches, routers, and other networking devices. The defendant is Arista Networks, a competitor founded by former Cisco employees.[33] The dispute centers on the operating system software for the networking hardware sold by both companies. Customers configure and operate these devices via a text-based command line interface ("CLI"). In Cisco's words: "The CLI is the user interface by which users of Cisco products communicate with the product in order to configure and manage the product."[34] For its copyright infringement claim, Cisco asserted that Arista infringed the user interfaces of the Cisco operating system software by copying hundreds of multiword commands, as well as multiword command hierarchies, modes and prompts, command responses and screen outputs, and help descriptions.[35]

While the district court described these interface elements in some detail,[36] the gist here should be familiar to anyone who has muddled through a similar command line interface on an early personal computer (MS-DOS) or a more recent Linux-based computer. Operating the Cisco switches requires using specific multiword com-

30. *Synopsys, Inc.*, 2016 WL 6158216, at *1. The patent claim was severed for separate trial.

31. Stipulation and Consent Judgment, Synopsys, Inc. v. ATopTech, Inc., No. 13-CV-02965 (N.D. Cal. filed Jun. 27, 2017) (ECF No. 967).

32. Complaint for Copyright and Patent Infringement, Cisco Sys. Inc. v. Arista Networks, Inc., No. 14-CV-053445344 (N.D. Cal. filed Dec. 5, 2014) (ECF No. 1).

33. Second Amended Complaint for Copyright and Patent Infringement at 1, Cisco Sys., Inc. Networks, Inc., No. 14-CV-05344 (N.D. Cal. filed July 23, 2015) (ECF No. 64).

34. *Id.* at 8.

35. Cisco Sys., Inc. v. Arista Networks, Inc., No. 14-CV-05344, Slip Opinion at 2–3 (N.D. Cal. May 10, 2017) (ECF No. 787).

36. *See id.*

mands in a particular syntax and format. Examples of multiword command expressions include "boot system," "show inventory," "area nssa translate type7 always," and "spanning-tree portfast bpdufilter default."[37] In order to configure and operate the Cisco switches, customers must employ the correct commands and expect the devices to return the relevant responses. In other words, this is the method of operation for Cisco switches.

In its competing products, Arista chose to implement support for many of the same multiword commands (approximately 500, according to Cisco) and other interface elements with which Cisco customers had become familiar over many years.[38] In various public statements, Arista specifically described how its support for these Cisco command line commands would help customers ease adoption and leverage their long-standing familiarity with Cisco commands.[39]

The dispute made its way to a jury trial in December 2016, where the jury returned a verdict of non-infringement in favor of Arista, finding the infringement excused by the scènes à faire doctrine.[40] The matter is, as of this writing, pending before the Court of Appeals for the Federal Circuit, where it will be heard in 2018 on the heels of Oracle's appeal from the fair use jury verdict in *Oracle v. Google*.[41]

Again, much could be said about the arguments of the parties, the various theories accepted and rejected by the court, and the details of the jury verdict. But for purposes of this discussion, the basic factual contours are most salient. Arista entered the market against an incumbent market leader whose customers had made substantial investments in learning the method for operating the incumbent's products. Arista wrote its own original software, but in order to mitigate switching costs for its potential customers, offered support for many of the same commands that Cisco's devices had long used. The similarities with *Lotus v. Borland*, *Oracle v. Google*, and *SAS v. WPL*, and *Synopsys v. ATopTech* are vivid.

D. GDC v. Dolby

The most recent of the new wave of interface cases came in the form of a declaratory judgment action filed by GDC Technology Ltd.

---

37. *Id.*

38. *See* Second Amended Complaint, *supra* note 33, at 3.

39. *Id.* at 2–3, 12–13.

40. Joe Mullin, *Arista Beats Cisco's $335M Copyright Claim with an Unusual Defense*, ARSTECHNICA (Dec. 14, 2016, 4:08 PM), https://arstechnica.com/tech-policy/2016/12/jury-clears-arista-of-ciscos-335m-copyright-claim/ [https://perma.cc/BD4C-EE8K].

41. The appeal is Cisco Sys., Inc. v. Arista Networks, Inc., No. 17-2145 (Fed. Cir. docketed June 13, 2017). Appellate jurisdiction is exclusive to the Federal Circuit, thanks to a pendant patent claim in the case.

against Dolby Laboratories, Inc. in April 2016.[42] According to the complaint, GDC and Dolby are competitors in the digital cinema industry.[43] GDC sells media servers to theater owners who have transitioned from film to digital cinema systems.[44] These media servers must interoperate with other systems provided by different vendors, including projector systems, sound systems, and theater management systems that coordinate the interoperation of these systems.[45] To orchestrate their coordinated functioning, these systems rely on commands that "typically take the form of a four-digit hexadecimal (two byte) code, embedded in a larger message header that tells the server that it is about to receive a message."[46] These command protocols are widely shared among all players in the industry, and in any event can easily be observed by reading data transmissions between different components of the theater system.[47]

Dolby's effort to upend this cooperative state of affairs triggered the litigation. According to the complaint, shortly after itself entering the media server market via acquisition, Dolby began asserting proprietary rights over its interconnection codes, telling theater owners that interconnecting GDC equipment with Dolby equipment would constitute an infringement of Dolby's intellectual property rights.[48] Dolby then followed up by sending a cease-and-desist letter to GDC, explicitly asserting copyright protection over the commands used to control Dolby devices.[49] GDC then filed the declaratory judgment action, insisting that "[t]he only element of Dolby's protocol that GDC uses is the set of messages/commands and corresponding hexadecimal interoperability codes,"[50] and asserting that the commands were not copyrightable or, in the alternative, that GDC's continued use would qualify as a fair use.[51]

Like the other "new wave" cases, this dispute bears strong similarities to prior interface cases. Once again, a leading incumbent asserted copyright in the commands necessary for the operation of its software. Prior customer sunk-cost investments (here, investments by theater owners in systems that include Dolby equipment and software) create lock-in at the expense of a competitor who seeks to enter the market. There was no allegation of copying of source code or other

42. Complaint, GDC Tech. Ltd., Inc. v. Dolby Labs., Inc., No. 16-CV-02459 (C.D. Cal. filed Apr. 11, 2016) (ECF No. 1).
43. *Id.* at 2.
44. *Id.* at 3.
45. *See id.* at 3–4.
46. *Id.* at 4.
47. *Id.*
48. *Id.* at 12–13.
49. *Id.* at 14.
50. *Id.* at 7.
51. *Id.* at 19.

literal elements of software; rather, the infringement claim was premised on a competitor creating original software that copies only the interface — the commands used by customers to operate the software.

Unlike the earlier cases, *GDC v. Dolby* settled before a judge or jury could rule on copyrightability or infringement.[52] In a terse press release, the parties disclosed merely that they "will grant each other licenses that will allow their respective theatre management systems to interoperate with the other party's digital-cinema servers."[53] This represents something of an about-face for GDC, which had asserted that the command protocols were not copyrightable. On that view, there was nothing for Dolby (and presumably GDC) to license. Nevertheless, in the wake of the settlement, presumably both Dolby and GDC remain free to assert copyright protection over their command protocols against other competitors.

III. LESSONS FROM THE NEW WAVE OF INTERFACE CASES

The new wave of software interface cases that have followed on the heels of *Oracle v. Google* fit the same pattern that characterized the previous cases described by Professor Menell. A market leader asserts a copyright claim over an interface — the method of operating the software — in an effort to bar a competitor from entering the market. The claim is not that the competitor is copying and redistributing the incumbent's source or object code. Instead, the allegation focuses on the competitor creating original software that reimplements some or all of the interface. The goal in each of these cases is to force competitors to create an entirely original interface as a precondition of market entry, thereby keeping switching costs high for the incumbent's existing customers. To return to the initial analogy, these are all examples of copyright being used to prevent a new entrant from using the equivalent of the steering wheel and pedals as the method for operating a car.

The assertion of copyright to hinder competitive entry is perhaps most vivid in *Lotus v. Borland*, *SAS v. WPL,* and *Cisco v. Arista*. In each of those cases, the defendant was entering the market with a directly competitive product, offering the incumbent's existing customers an easy way to bring their existing macros, programs, or scripts over with them. In *Oracle v. Google*, Android and Java were not competing products, but nevertheless the theme of switching costs is clear. Java developers had independently developed fluency in Java's

52. Stipulation of Dismissal Pursuant to Fed. R. Civ. P. 41(a), GDC Tech. Ltd. v. Dolby Labs., Inc., No. 16-CV-02459 (C.D. Cal. Oct. 26, 2016) (ECF No. 26).

53. Film J. Int'l, *GDC and Dolby Resolve Litigation* (Nov 1, 2016), http://www.filmjournal.com/news/gdc-and-dolby-resolve-litigation [https://perma.cc/R9RU-59TF].

API declarations. They also had already written software in Java, code that would be difficult to repurpose if it had to be entirely rewritten. In Professor Menell's words: "Using some of the Java APIs provided a bridge for the millions of Java programmers."[54] In this case, the developers were the customers, and Oracle's assertion of copyright over the Java API declarations effectively increased switching costs for developers who wanted to try a different programming ecosystem.

GDC v. Dolby and *Synopsys v. ATopTech* also fit the same pattern. In *GDC v. Dolby*, the assertion of copyright over the command protocol was a claim limited solely to the interface — Dolby never suggested that GDC had copied and incorporated Dolby's software into GDC's products. By using copyright law to create artificial incompatibility between Dolby and GDC products, customers would be forced to choose one or the other, thereby increasing switching costs for customers who wanted to defect from Dolby to GDC, or mix-and-match products from both vendors. And while some of the facts in *Synopsys v. ATopTech* are obscured by redactions to protect the proprietary information of the parties, the story that emerges is also one of a leading incumbent asserting copyright over an interface — in that case, input and output formats — in an effort to prevent customers from defecting to the competitor's interoperable products.

As Professor Menell explains, it is difficult to see why copyright law should tolerate these kinds of claims. I will not reiterate the statutory and jurisprudential arguments that Professor Menell thoroughly covers. On that score, two simple points are enough to persuade me. First, § 102(b) of the Copyright Act expressly provides that "[i]n no case does copyright protection for an original work of authorship extend to any . . . method of operation . . . regardless of the form in which it is described, explained, illustrated, or embodied in such work."[55] Second, as recognized in *Lotus v. Borland*, this statutory command applies equally to software.[56] Accordingly, the interface used to control a software program cannot qualify for copyright protection, insofar as it serves as the "method of operation" for the software. The Federal Circuit was mistaken to the extent it held otherwise in its copyrightability ruling in *Oracle v. Google*.

Professor Menell also ably covers the economic and policy rationales that support his view. Extending copyright protection to software interfaces is not necessary to provide adequate incentives for software developers.[57] Copyright law will continue to protect software

54. Menell, *supra* note 2, at 470.

55. 17 U.S.C. § 102(b) (2012).

56. Lotus Dev. Corp. v. Borland Int'l, Inc., 49 F.3d 807, 815–18 (1st Cir. 1995), *aff'd by an equally divided Court*, 516 U.S. 233 (1996).

57. Menell, *supra* note 2, at 464 ("[F]unctional features of computer software and machines fall within the patent system's domain. The importance of interoperability and com-

from wholesale, piratical duplication. Patent law may protect interfaces, if they qualify under patent law's more stringent requirements. And software companies will retain the benefits of trade secret protection, contract law, and technological protection measures. Together, these will frequently yield a significant first-mover advantage, as competitors will be required to study, document, and independently reimplement interfaces before entering the market. Furthermore, as explained by Professor Menell, a significant portion of the software industry has moved to nonproprietary and cloud-based business models that are protected from much of "the appropriability problem"[58] that copyright is designed to solve. Granting copyright protection to interfaces, in contrast, conveys a century of legal protection against new market entrants who want to "build a bridge"[59] for customers interested in trying their products.

Returning to the fundamental purpose of copyright, moreover, developers seem to have sufficient incentives to develop interfaces, so an exclusive right appears unnecessary. After all, the incentive to create a method of operating something is inherent in the incentive to create the thing itself. It would be a strange piece of software indeed that offered no interface, no method of operation, just as it would be strange to find a car maker building an automobile with no way for a driver to operate it.[60]

In the end, the question that the new wave of software interface cases raises is the same one identified by Judge Boudin in *Lotus v. Borland*:

> 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.[61]

---

patibility bring trademark protection into play. In addition, software can often be protected through trade secret law. Developers can hide their programming by only releasing object code versions to the public.").

58. *Id.* at 463 ("To a significant extent, platform developers have partially addressed the appropriability problem associated with software development through the formation of collaborative clubs.").

59. *Id.* at 458 ("Another strategy is to build a convenient bridge over which consumers can easily migrate to and become accustomed to a new platform.").

60. What about self-driving cars? To the extent self-driving cars may soon dispense with the steering wheel and pedals, software will provide the new interface, raising all the concerns discussed here. If consumers become accustomed to a particular set of spoken word commands to control their self-driving cars, will those commands be subject to copyright protection, requiring drivers to learn an entirely new command vocabulary in order to try a different make of self-driving car?

61. 49 F.3d at 821.

Professor Menell restates this in economic terms as the "network externality dilemma."[62] It stands to reason that empowering an incumbent market leader in a market characterized by network effects to recoup not only the value of its initial investment, but also the independent investments of its customers, threatens exactly the kind of monopolistic deadweight loss that Professor Menell describes.[63]

IV. WHY AFFIRMATIVE DEFENSES ARE NOT ENOUGH

As discussed earlier, there are those who agree that copyright protection for software interfaces can have socially detrimental consequences, but who believe that copyright's existing affirmative defenses can adequately address these concerns. Professor Menell addresses this at the end of his article, opining that "the fair use trial [in *Oracle v. Google*] was a massive waste of time, party resources, and judicial resources."[64] He also emphasizes that sending these cases to jury verdicts is extremely expensive and does not deliver the legal clarity that software developers and investors need in order to make "the difficult, time-sensitive decisions involved in designing products and platforms."[65]

With respect to Professor Menell's point regarding expense, skeptics may point out that recognizing interfaces as unprotectable "methods of operation" will not always spare litigants the burden of a trial. After all, in *Oracle v. Google*, the district court initially did not rule on the protectability of the interfaces in question, but instead instructed the jury to assume copyrightability, and the jury returned a hung verdict on fair use.[66] Nevertheless, it seems likely that filtering out methods of operation as unprotectable should, in most cases, result in earlier (and cheaper) outcomes for litigants. The only relevant inquiry under § 102(b) should be whether the interface constitutes a "method of operation." While the question may not be free from doubt in all cases, it certainly will not require the kind of broad ranging factual inquiries required for a fair use or scènes à faire determination. In many cases, the matter could be resolved at summary judgment, making a jury trial unnecessary.

Professor Menell is likewise on solid ground when he notes that legal clarity suffers if the questions raised by protecting interfaces are left to copyright's affirmative defenses. One need only look at the

62. Menell, *supra* note 2, at 458 ("Intellectual property protection both contributes to and alleviates the network externality dilemma.").

63. *Id.* at 454 ("Monopolistic exploitation distorts market pricing in the short run and can significantly affect entry and cumulative innovation over longer time horizons.").

64. *Id.* at 471.

65. *Id.* at 472.

66. Final Charge to the Jury (Phase One) and Special Verdict Form, Oracle Am., Inc. v. Google Inc., 2016 WL 4368346 (N.D. Cal. May 26, 2016) (No. C 10-03561 WHA).

interface cases where affirmative defenses were litigated to see the lack of predictability that results. In *Oracle v. Google*, for example, the defendant prevailed on a fair use defense (but only after trying the issue twice), while in *Lotus v. Borland* and *Synopsys v. ATopTech*, the defendants' fair use defenses were unavailing.[67] In *Cisco v. Arista*, the defendant's fair use arguments failed, but its scènes à faire argument carried the day.[68] Of course, in every case, the outcome will turn on the facts. But in cases that turn on affirmative defenses like fair use and scènes à faire, the outcomes turn on facts that are specific to the particular product and industry, rather than on any generalizable principles regarding the protectability of interfaces that can guide other software developers.

Some may conclude that this is a feature, rather than a bug, in that the question of protectability should turn on the particulars of the product at issue; every interface case is unique. This argument, of course, reinforces Professor Menell's point that these cases will then provide very little legal clarity for others. But those who nevertheless prefer the close-up, particularized treatment afforded by affirmative defenses must answer another question: do the affirmative defenses like fair use and scènes à faire lead courts and juries to ask the right questions? As discussed by Professor Menell and reinforced by the recent cases discussed above, the crucial social welfare questions raised in software interface cases revolve around network effects, switching costs, and market entry. Are copyright's existing affirmative defenses the best tools for addressing these questions?

Probably not. In Professor Menell's words: "The fair use doctrine is an especially poor vehicle for resolving API copyright disputes."[69] In some interface disputes, the fair use doctrine may leave room for the relevant policy considerations. In *Oracle v. Google*, for example, the Android operating system was transformative in nature (using a tiny portion of the Java API interface as part of a much larger, more ambitious operating system in a different market) and did not compete with the copyrighted work, the Java SE software. These characteristics not only track the first and fourth fair use factors,[70] but also point up the switching costs faced by Java developers and the innovation unlocked thanks to Google "building a bridge" between the Java and

67. *See Lotus*, 49 F.3d at 812; Jury Verdict at 1, Synopsys, Inc. v. ATopTech, Inc., No. 13-CV-02965 (N.D. Cal. filed Mar. 10, 2016) (ECF No. 687).

68. *See* Mullin, *supra* note 40.

69. Menell, *supra* note 2, at 471.

70. The first fair use factor addresses the "purpose and character of the use" (including whether the use is transformative) and the fourth fair use factor addresses "the effect of the use upon the potential market for or value of the copyrighted work" used. 17 U.S.C. § 107 (2012).

Android platforms.[71] But fair use did not carry the day for Arista in its battle over the use of Cisco's multiword commands. And what about cases like *SAS v. WPL* or *GDC v. Dolby*, where the defendants are direct competitors and their products do not easily fit the category of "transformative"? In these cases, the switching cost and network effects questions loom equally large, but the fair use factors seem to deemphasize these impacts on customers in favor of a focus on potential harms to the copyright owner.

Scènes à faire also does not seem to reliably grapple with the social welfare impacts of protection for interfaces. In *Cisco v. Arista*, it seemed that the scènes à faire doctrine may have encompassed concerns about switching costs and network effects, insofar as Arista introduced evidence demonstrating that many of the Cisco commands stemmed from older conventions widely used by networking engineers, leaving the options for alternatives limited.[72] These facts indirectly relate to the notion that Cisco should not be entitled to use preexisting conventions to block Arista's entry into the market. Had the litigation in *GDC v. Dolby* moved ahead, scènes à faire might have similarly been adequate, insofar as the commands were short and largely defined by prior conventions in the theater systems market. But the relationship between the economic concerns raised in these cases — switching costs and network effects — and scènes à faire is, at best, indirect. And cases like *SAS v. WPL* show how scènes à faire might fall short in vindicating the economic policies that Professor Menell identifies. In that case, the SAS input and output formats were less the product of well-established convention and limited options (the concerns of scènes à faire), but rather the product of the SAS language and the leading position that SAS had established over many years.

The drawbacks posed by reliance on affirmative defenses to resolve copyright disputes over software interfaces, taken together, may have yet another unfortunate consequence: tilting the playing field toward big, well-resourced companies. While federal civil litigation is always expensive, a reliance on affirmative defenses will make it harder for defendants to prevail at an early stage. As described above, forcing defendants to endure the entire process of discovery and trial before a jury will make interface disputes more expensive, drawn out, and unpredictable. This, in turn, will favor those companies who can weather the expense and risks of unpredictable, drawn out federal litigation. These realities will likely reduce the number of companies

71. Menell, *supra* note 2, at 470 ("Using some of the Java APIs provided a bridge for the millions of Java programmers.").

72. *See* Def. Arista Networks, Inc.'s Notice of Mot. and Mot. for Partial Summary Judgment at 9–10, Cisco Sys. Inc. v. Arista Networks, Inc., No. 14-CV-05344 (N.D. Cal. filed June 30, 2016) (ECF No. 329) (citing expert reports).

willing to risk disruptive market entry against well-heeled incumbents with sizeable litigation war chests. Seen in this light, perhaps it is no surprise that Dolby and GDC settled their dispute with a cross-licensing arrangement: they have achieved peace while sending a strong message to any new market entrant.

V. CONCLUSION

In the final analysis, the new wave of interface cases fit into the historical pattern identified by Professor Menell and bear out the wisdom of his ultimate conclusions: "Leaving API design specifications outside of copyright protection enables entrepreneurs seeking to improve on successful platforms to build bridges for users and programmers. This avoids excess inertia and accommodates creative destruction and evolution in those areas where the proprietor of the standard platform lacks patent protection."[73]

73. Menell, *supra* note 2, at 468.