The Past, Present and Future of Software Copyright Interoperability Rules in the European Union and United States

Pamela Samuelson
Richard M. Sherman Distinguished Professor of Law, University of California, Berkeley

Copyright; EU law; Interoperability; Software; United States

The protectability (or not) of computer program interfaces, the legality of reverse engineering of program code to extract interface information and the reimplementation of interfaces in complementary or competing programs was deeply controversial in the late 1980s and early 1990s. For the past 20 years, copyright law in both the European Union and United States has been favourably disposed towards treating interfaces necessary to achieving interoperability as unprotectable elements of programs and towards reverse engineering for a legitimate purpose such as discerning interface information. Controversies over interfaces and interoperability have, however, not ceased. The Court of Justice of the European Union is now considering an important case, SAS Institute, Inc v World Programming Ltd, which calls for an interpretation of the Council Directive 91/250 on the legal protection of computer programs concerning the protectability of interfaces designed to enable a competing program to interoperate with existing programs.

Introduction

It was far from inevitable in either the European Union or the United States during the 1980s and early 1990s that copyright in computer programs would be unavailable to interfaces necessary for achieving interoperability. Interfaces are, after all, complex and detailed information resources embodied in programs that may require skill, judgment and creativity to design, thus seemingly satisfying copyright’s originality standard.1

Several leading US firms—notably IBM Corp. and Microsoft—were allied in that era in arguing on both sides of the Atlantic that copyright protection in programs should extend to interfaces because they were creative parts of program structure.2 Some EU firms, including Olivetti and Bull, along with some US firms, such as Sun Microsystems and NCR, formed the European Committee for Interoperable Systems (ECIS) to persuade the European Commission that interoperability was so critical to competition and ongoing innovation in the software industry that copyright protection should be unavailable to interfaces and that interfaces should be treated for copyright purposes as ideas.3

The first part of this article explains the state of play about software interoperability and copyright law in the late 1980s, when it seemed as if copyright protection might be available for interfaces and unauthorised interoperability would be disfavoured. The second part discusses how the state of play on interfaces and interoperability changed in the early 1990s. Although the European Union and United States took quite different paths to get there, their laws converged on the two most important copyright issues relevant to interoperability: first, that interfaces necessary to interoperability may be unprotectable by copyright law, and secondly, that reverse engineering of computer programs, insofar as it is necessary to discerning interface information, does not infringe software copyrights.4

While the law on software copyright interoperability issues seems quite settled on both sides of the Atlantic, the third part points to some litigation involving software copyrights that might undermine the consensus in favour of interoperability unless courts carefully attend to policy rationales that have led to the consensus in favour of interoperability. Especially important to watch is SAS Institute, Inc v World Programming Ltd.5 The UK High Court judge referred several interoperability-related questions posed in that case to the Court of Justice of the European Union (CJEU) for its interpretation of EU software copyright law.6

The pre-history of software interoperability in copyright law

In the late 1970s and well into the 1980s, it was not even inevitable that copyright protection would be available for computer programs. The World Intellectual Property Organization (WIPO) initially proposed a sui generis (of

1 This article is a derivative work of the keynote speech given at an event celebrating the 21st anniversary of the European Committee for Interoperable Systems (ECIS) held on December 1, 2011 in Brussels, Belgium. I wish to thank ECIS for the honour of giving this lecture. Thomas Vinje for his support and encouragement, and Ashwin van Rooijen for comments on a draft of this article.
2 See, e.g., Ashwin van Rooijen, The Software Interface between Copyright and Competition: A Legal Analysis of Interoperability in Computer Programs (Kluwer Law International, 2010), pp.7–47 (explaining software interfaces as they affect interoperability and originality issues).
its own kind) form of IP protection for software. Japan was among the nations favourably inclined toward sui generis software IP legislation. In the late 1980s, the European Commission issued a Green Paper about legal protection for computer programs which considered a sui generis approach, along with copyright law, as a policy option.

The United States had, however, become firmly committed to copyright protection for computer programs by 1980. Even so, this commitment had actually been fairly slow to develop. In the mid-1960s, the US Copyright Office had only been willing to accept registration of computer programs under its “rule of myth”. The Office’s doubts concerned whether object code forms of programs could be protected by copyright law because of their functionality and unobservability. Similar questions about the copyrightability of computer programs were raised in the legislative history of the Copyright Act of 1976 (1976 Act).

Because a controversial set of new technology issues, including the suitability of copyright for protecting software, were holding up enactment of what became the 1976 Act, the US Congress in 1974 created the National Commission on New Technological Uses of Copyrighted Works (widely known as CONTU) to make recommendations about them. CONTU’s Final Report endorsed copyright protection for computer programs and optimistically predicted that the idea-expression distinction would provide a sound mechanism for balancing competing interests of right holders and the public. It regarded the 1976 Act as providing copyright protection to programs insofar as they were “original works of authorship fixed in a tangible medium of expression”. Congress dispelled whatever ambiguity might have existed about the copyrightability of programs in 1980 when it passed legislation to implement CONTU’s recommendations.

Unfortunately, CONTU did not consider any interoperability-related issues in its report. Nor did it discuss the copyright status of computer programming languages or program behaviour, although these aspects of programs were arguably excluded from copyright protection as “procedures, processes, systems, [and] methods of operation” that the 1976 Act puts outside the scope of copyright protection in works of authorship. A Congressional report indicated that this exclusion had been added to the statute in order to ensure that the scope of copyright protection in programs would not extend too far.

The first US appellate court decision to consider interoperability issues was Apple Computer, Inc v Franklin Computer Corp in 1983, which cast doubt on the viability of compatibility defences in software cases. Franklin made computers that were clones of the Apple II platform for which many programs had been written. Franklin achieved compatibility with programs written for the Apple II by installing on its computers exact copies of the Apple II operating systems (OS) programs.

Franklin argued that this was necessary in order for its computers to be compatible with programs written for the Apple II platform, rendering the ideas in the Apple II OS as having “merged” with the programs’ expression.

Under the merger doctrine of US copyright law, this would cause the expression to be treated as an unprotectable idea. The Third Circuit Court of Appeals rejected this defence, saying:

“Franklin may wish to achieve total compatibility with [Apple II applications], but that is a commercial and competitive objective which does not enter into the somewhat metaphysical issue of whether particular ideas and expressions have merged.”

While the court was right to consider Franklin an infringer, it should instead have rejected Franklin’s compatibility defence because Franklin hadn’t even tried to rewrite the Apple OS programs to reimplement the interfaces in independently written code.

Three years later the Third Circuit rendered another opinion that seemed to bode ill for compatibility defences in Whelan Associates, Inc v Jaslow Dental Laboratory, Inc. Jaslow had hired Whelan to develop a computer program to automate common business operations of dental laboratories. He wanted to use this program in his own firm, but also to partner with Whelan to sell it to other dental labs. He was the domain expert about dental lab business operations, but as the project wore on, he became a programmer. Jaslow recognised the likely market demand for an IBM-PC program to perform the same operations as the program that Whelan had developed for IBM mainframe computers. Jaslow’s IBM-PC program was written in a different programming language and used different algorithms from Whelan’s,

---

3 See Green Paper on Copyright and the Challenge of Technology—Copyright Issues Requiring Immediate Action, Document COM(88) 172 final (June 7, 1988).
8 17 USC § 102(a).
10 17 USC § 102(c).
12 Apple Computer, Inc v Franklin Computer Corp 714 F. 2d 1240 (3d Cir. 1983).
13 Apple Computer 714 F. 2d 1240, 1241-1252 (3d Cir. 1983).
14 Apple Computer 714 F. 2d 1240, 125 (3d Cir. 1983).
15 Whelan Associates, Inc v Jaslow Dental Laboratory, Inc. 797 F. 2d 1222 (3d Cir. 1986).
but the overall structure of Jaslow’s program and some file and data formats were the same. Moreover, some subroutines of Jaslow’s program were substantially similar in operation to those of Whelan’s program.

Whelan sued Jaslow for infringement, claiming that the “structure, sequence, and organisation” (SSO) of her program were protectable by copyright law and Jaslow had copied them. Because computer programs were literary works under the 1976 Act definition of the term, Whelan argued that the SSO of her program should be protectable expression, just as was the SSO of conventional literary works such as dramatic plays and novels. Whelan further argued that without copyright protection for SSO, as well as for the “look and feel” (that is, the behaviour), of programs, there would be too little incentive to invest in developing software. Jaslow’s main defence was that software copyrights only protected against exact copying of program code, and since he had written his own program, he was not an infringer.

The Third Circuit agreed that the structural similarities between Whelan’s and Jaslow’s programs were a proper basis for imposing copyright liability, broadly endorsing copyright protection for computer program SSO, as well as for look and feel. It also announced a test for software copyright infringement under which the general purpose or function of a program was an unprotectable idea; more detailed elements of programs, such as SSO and look and feel, would be protectable expression unless there was only one or a very small number of ways to design a program of that sort, in which case idea and expression would be considered merged, and protection withheld from the merged idea/expression.

Although Whelan did not directly involve interoperability issues, it had implications for future cases involving interface issues. For one thing, interfaces can be considered part of program SSO, so Whelan arguably implied that copying this kind of program SSO would infringe. Secondly, Whelan also considered the statutory exclusions of methods and processes from the scope of copyright to be no more than a restatement of the idea/expression distinction, thereby implying that detailed methods and processes, such as interfaces, might be expression. Thirdly, the court in Whelan was willing to infer from similarities in the ways that programs functioned that there were underlying structural similarities in programs, which the defendant might have copied from the plaintiff. By predicating infringement liability on similarities in the way that program subroutines functioned, Whelan set a precedent that made emulation of program functionality risky, even when it was important to achieving interoperability.

In the late 1980s and early 1990s, Whelan was highly controversial in the United States. Although it was followed in several subsequent software copyright cases, it was heavily criticised in the law review literature, and some courts found it unpersuasive.

Despite the controversy over Whelan in the United States, that decision had some influence in the European Union. It was a key precedent on which advocates for strong copyright protection for interfaces relied in arguing to the European Commission (EC) in 1989 that its proposed directive on the legal protection for computer programs should protect program SSO, including interfaces.

The initial draft of that Directive seemed consistent with this point of view, for it was quite vague about the extent to which interfaces might be unprotectable ideas. This greatly distressed companies that were in the business of developing products compatible with those of industry leaders such as IBM. High protectionists firms noted that interface information was often available through licensing, which meant, in their view, that no copyright exception was needed to promote interoperability.

The high protectionists also insisted that decompilation and other modes of reverse engineering program code—other than “black-box testing”, which consists of running a program through various operations to observe how its functions were performed—should be treated as copyright infringements because this process involved making copies of program code, which implicated the reproduction right. Although the US case law on the reverse engineering of program code was unresolved, the high protectionists argued that this kind of reverse engineering was illegal under US law, and that the European Union should conform its directive to this rule.

23 Ibid., 797 F.2d at 1228 (3d Cir. 1986).
24 Ibid., 797 F.2d at 1231 (3d Cir. 1986).
25 Ibid., 797 F.2d at 1222, 1223 (3d Cir. 1986).
26 Ibid., 797 F.2d at 1222–1224, 1228, 1229–1228 (3d Cir. 1986).
27 Ibid., 797 F.2d at 1236 (3d Cir. 1986).
28 Ibid., 797 F.2d at 1236 (3d Cir. 1986).
29 Ibid., 797 F.2d at 1236 (3d Cir. 1986).
30 Ibid., 797 F.2d at 1244 (3d Cir. 1986).
31 Ibid., 797 F.2d at 1245–1246 (3d Cir. 1986).
32 The need to emulate functional behaviour in order to achieve interoperability became important in the Lotus v. Borland case discussed in fn. 69–72 below and accompanying text, as well as in SAS v. WPI, discussed below in fn. 74–85 and accompanying text.
33 See, e.g., Johnson Controls, Inc v Phoenix Control Sys., Inc 886 F. 2d 1173, 1175 (9th Cir. 1989).
34 See, e.g., Computer Associates Int'l v Altai, Inc 982 F. 2d 693, 705 (2d Cir. 1993) (citing numerous articles critical of Whelan).
35 See, e.g., Plains Cotton Co-op v Goodpaster Computer Serv., Inc 807 F. 2d 1256, 1263 (5th Cir. 1987).
40 See, e.g., Hvidt Corp v Quad Software Ltd 847 F. 2d 255 (5th Cir. 1988) (suggesting this was lawful); Hubco Data Products Corp v Management Assistance, Inc 219 U.S.P.Q. 450 (D. Idaho 1983) (casting doubt on the lawfulness of reverse engineering of program code).
Because of the US precedents and the lobbying clout of leading US firms, it was an uphill battle for proponents of interoperability to persuade the EC that interfaces should not be protectable SSO under copyright law and that reverse engineering should be lawful when it was necessary to achieving interoperability. And yet this happened in the early 1990s.

The emergence of pro-interoperability rules for software copyright law

The legislative history of the Council Directive on the legal protection of computer programs (Software Directive) has been told in considerable detail and very well by numerous European commentators. Rather than retell this story, this article will simply note that the proponents of pro-interoperability rules succeeded in persuading EU policy-makers that as long as programmers wrote independent software that did not copy another firm’s program code or expressive parts of the internal structure of programs, there would be more competition and more innovation in the software industry if it was lawful for programs to interoperate with existing programs.

The Software Directive does not categorically exclude interfaces from the scope of copyright’s protection, although it anticipates that interfaces may be among the unprotectable elements of programs as ideas or principles. It recognises that “the function of a computer program is to communicate and work together with other components of a computer system and with users”, and that:

“[A] logical and, where appropriate, physical interconnection and interaction is required to permit all elements of software and hardware to work with other software and hardware and with users in all the ways in which they are intended to function.”

Interfaces are defined as “the parts of the program which provide for such interconnection and interaction”, and interoperability as “the ability to exchange information and mutually to use the information which has been exchanged”.

It is fair to conclude from these definitions and the strong emphasis on interfaces and interoperability in the Software Directive that it was intended to promote reuses of interfaces essential to interoperability. This policy also plainly underlies a provision of the Software Directive that forbids decompilation of computer program code except insofar as it is “indispensable to obtain the information necessary to achieve the interoperability of an independently created computer program with other programs”.

In the year after the European Union adopted the Software Directive, substantially the same rules about program interfaces and reverse engineering emerged in the United States, although this was achieved through judicial interpretation rather than legislatively. Computer Associates Int’l Inc v Altai, Inc interpreted US copyright law as not extending protection to interfaces necessary to interoperability. Sega Enterprises Ltd v Accolade, Inc ruled that reverse engineering of computer program code for a legitimate purpose such as extracting interface information from another firm’s software was a fair and non-infringing use of that code. Neither decision referred to the Software Directive; they relied solely on US copyright law and policy. Because some European readers may be less familiar with these precedents than US readers, this article will discuss the courts’ analyses in these two important cases.

Computer Associates (CA) and Altai were competitors in the business of producing software to perform scheduling tasks for IBM mainframe computers. Because IBM did not use the same OS on all of its computers, any scheduling program designed to run on more than one IBM OS had to conform to the interfaces that each IBM OS required. In preparing to update its scheduling software so that it would operate on more than one IBM machine, Altai hired a former CA employee to work on this project.

Unbeknown to Altai, this person directly copied some code from CA’s software for the update of Altai’s software. When CA sued Altai for infringement on account of this copying, Altai immediately stopped shipping the infringing code. It then assigned one set of programmers to study the infringing code to extract interface information and another set of programmers to reimplement the interfaces in independently written code. CA charged that Altai was still an infringer because of similarities in the SSO of the two programs, particularly as to its parameter lists.

In its Altai decision, the Second Circuit Court of Appeals took issue with Whelan’s overbroad view about the protectability of program SSO. Although accepting
that some structural elements of programs might be protectable expression, the Second Circuit recognized that computer programs were “essentially utilitarian” works, which “complicates the task of distilling idea from expression.” The appropriate “doctrinal starting point” for analyzing the scope of copyright in utilitarian works was the venerable Supreme Court precedent, *Baker v Selden*, which held that copyright protection was unavailable to a useful art such as the novel bookkeeping system, described in the plaintiff’s work; copyright only protected the author’s *explanation* of the system, not the *methods of operation* the system employed. In keeping with *Baker*, US courts have been careful to ensure that utilitarian methods and processes described in copyrighted works are not protected by copyright in a text discussing them. The Second Circuit concluded that *Whelan* “relies too heavily on metaphysical distinctions and does not place enough emphasis on practical considerations”.

Among those practical considerations were:

“extrinsic considerations such as (1) the mechanical specifications of the computer on which a particular program is intended to run [and] (2) compatibility requirements of other programs with which a program is intended to run”

, as well as “demands of the industry being serviced” and “elements taken from the public domain”.

These considerations were significant in *Altai* because both CA’s and Altai’s programs needed to be compatible with IBM OS programs as well as to provide substantially the same services to attract customers. After excluding similarities due to these factors, the similarities in the two programs were too insubstantial to support a finding of infringement.

A few months later, the Ninth Circuit Court of Appeals ruled in *Sega v Accolade* that compiling or disassembling a computer program for a legitimate purpose, such as to get access to interface information, did not infringe copyright. Although Accolade had produced videogames to run on other platforms, it wanted to adapt these games to run on Sega’s popular game console.

Because Sega had not published information about its interfaces and the terms on which it was willing to license this information were highly restrictive, Accolade had no choice but to reverse engineer Sega programs “to discover the functional requirements for compatibility with the Genesis console — aspects of Sega’s programs that are not protected by copyright”.

The court observed that:

“Accolade did not seek to avoid paying a customarily charged fee for use of those procedures, nor did it simply copy Sega’s code; rather, it wrote its own procedures based on what it had learned through disassembly.”

These factors contributed to the court’s decision that the copies made in the course of Accolade’s reverse engineering were non-infringing fair uses. This ruling relied on *Altai* as having established that external factors, such as the need to be compatible with other software or hardware, limited the design choices of subsequent programmers.

US courts typically consider four factors in determining whether a challenged use is fair: the purpose of the defendant’s use, the nature of the copyrighted work, the amount copied, and the potential for harm to markets for the plaintiff’s work.

The Ninth Circuit decided that Accolade’s purpose in making copies of Sega’s programs was to study it to extract information necessary to achieve interoperability with the Sega platform. The nature of the work factor was significant because object code cannot be read in the same manner as conventional literary works; it is necessary to disassemble or decompile this code to convert it to a human-readable form to extract interface information. Although Accolade had copied whole Sega programs to discern interface information, the necessity of this copying mitigated this factor. The court perceived no copyright-relevant harm to the market for Sega’s works because Accolade’s games were non-infringing works that competed in the market with Sega’s in a manner that copyright law is designed to encourage.

The Ninth Circuit also considered that:

“If disassembly of copyrighted object code is per se an unfair use, the owner of the copyright gains a de facto monopoly over the functional aspects of his work—aspects that were expressly denied copyright protection under 17 U.S.C. § 102(b).”

---

51 *Altai* 982 F. 2d 693, 704 (2d Cir. 1992).
54 *Altai* 982 F. 2d 693, 706 (2d Cir. 1992).
55 *Altai* 982 F. 2d 693, 709-710 (2d Cir. 1992).
56 *Altai* 982 F. 2d 693, 710 (2d Cir. 1992).
57 *Altai* 982 F. 2d 693, 715 (2d Cir. 1992).
58 *Altai* 982 F. 2d 693, 714-715 (2d Cir. 1992).
59 *Sega* 977 F. 2d 1510 (9th Cir. 1992).
60 *Sega* 977 F. 2d 1510, 1522-1523 (9th Cir. 1992).
61 *Sega* 977 F. 2d 1510, 1522-1523 (9th Cir. 1992). The Ninth Circuit agreed with Altai’s criticism of the *Whelan* decision (at 1525).
62 17 USC §107 (setting forth fair use factors).
63 The Ninth Circuit’s fair use analysis appears in *Sega* 977 F. 2d 1510, 1522-1525 (9th Cir. 1992).
64 *Sega* 977 F. 2d 1510, 1526 (9th Cir. 1992).
To get exclusive rights over functional aspects of programs, the court thought that Sega needed to get a patent and satisfy the more rigorous standards of that IP regime.67

Altai and Sega are the two most significant software copyright cases in the United States. Their holdings have been followed in subsequent case law and accepted as sound precedents in the legal commentary about interoperability-related issues.48

The next most significant US software copyright case after Altai and Sega is Lotus Development Corp v Borland Int'l, Inc.50 Borland is partly significant because it was the third major appellate court decision that rejected the Whelan approach to analysing software copyright issues, thereby putting the final nail in the coffin of this overbroad interpretation of US copyright law. Borland's rejection of a claim of copyright infringement is significant because similarities between the Lotus and Borland programs were due to their use of the same detailed methods of operating a computer to perform spreadsheet functions,19 thereby giving substance to the statutory exclusion of methods of operation from the scope of copyright protection. Borland has also attracted support in subsequent US case law and commentary.32

New challenges to interoperability defences in the European Union and United States

Adoption of the Software Directive in 1991 brought considerable stability to the legal protections for computer programs in the European Union, just as Altai and Sega brought similar stability to US software copyright law. The pro-interoperability rules of both jurisdictions have contributed to the phenomenal growth of this industry, allowing new entrants to the market for complementary and competing products.31 Consumers have benefited considerably from the availability of a wide range of interoperable information technologies.

This once hard-fought peace and harmony on interoperability issues is not, however, completely assured on either side of the Atlantic because of some cases now pending in courts in the European Union and United States. The most important such case is SAS v WPL, now under consideration by the CJEU for its interpretation of the Software Directive.34

That lawsuit came about because WPL developed a program that emulates the functionality of SAS's widely used statistical analysis program.35 An important feature of the SAS program is the programming language that SAS created to enable users to construct programs (known as "scripts") in that language to carry out specialised statistical operations on the SAS platform. WPL recognised that there was a potential market demand for an alternative software platform on which users could execute SAS's language-scripts.

In preparing its program to compete with SAS, WPL relied on two principal sources of information: first, it studied the manner in which the SAS program performed various operations through "black box" tests using copies of SAS programs WPL purchased in the mass market, and secondly, it read SAS manuals which describe in detail what the SAS software does and how the SAS language could be used to construct scripts. WPL had no access to SAS source code or to internal design documentation of the program, either directly or through decompilation.36

WPL's program emulated the functional behaviour of the SAS program. It did so to allow users who had constructed scripts in the SAS language to execute them on the WPL platform. WPL's program was, in effect, a drop-in replacement for the SAS program; that is, it accepted the same inputs and produced the same outputs as the SAS program. It had to utilise the same data formats as the SAS program in order for scripts written in the SAS language to interoperate with the WPL platform.

SAS sued WPL for copyright infringement, claiming that it was infringement to copy the behaviour of the SAS program, the SAS programming language and the formats for entering data into the SAS program. SAS argued that these were its intellectual creations, and hence original expressions to which copyright protection extended.

The UK High Court ruled that the functional behaviour of the SAS program was not protectable expression under the Software Directive because it embodied methods of operation that were unprotectable by copyright law under

67 Sega 977 F. 2d 1510, 1526 (9th Cir. 1992).
68 See, e.g., Sony Computer Entertainment, Inc v Connectix Corp 203 F. 3d 596 (9th Cir. 2000); Bateman v Mnemonics, Inc 79 F. 3d 1532 (11th Cir. 1996); Gates Rubber Co v Bando Chem. Indus. 9 F. 3d 823 (10th Cir. 1993); Band and Katoh, Interfaces on Trial 2.0 (2011), pp.31-67.
69 Lotus Development Corp v Borland Int'l, Inc 49 F. 3d 807 (1st Cir. 1995).
70 The First Circuit's Borland decision did not cite or discuss the Whelan case; however, Whelan had been influential with the District Court. See Summervue, "Why Copyright Excludes Systems and Processors From the Scope of its Protection" (2007) 85 Texas L. Rev. 1921, 1962-1973 (relating this history in greater detail).
71 Borland 49 F. 3d 807, 818: 819 (1st Cir. 1995).
72 See, e.g., Bateman v Mnemonics, Inc 79 F. 3d 1532 (11th Cir. 1996); Band and Katoh, Interfaces on Trial 2.0 (2011), pp.36-37.
74 See, e.g., George Arreto, Judge Rapontre, Report for the Hearing, SAS Institute, Inc v World Programming Ltd (C-406-10), reference for a preliminary ruling pursuant to art.267 TFEU from the High Court of Justice (Chancery Division) (United Kingdom) (describing the issues before the CJEU and the parties' arguments about the Software Directive).
international treaty provisions with which the Directive should be consistent. The court further ruled that the SAS programming language was outside the scope of protection that copyright law provided to programs as literary works and that the data formats at issue were indispensable to achieving interoperability and hence also outside the scope of protection under the Software Directive. Yet because the CJEU had not yet rendered a definitive interpretation of the Software Directive, the UK High Court thought that it would be best to refer specific interpretive questions to the court.

In an article published last month, two European colleagues and I have asserted that the UK High Court properly interpreted the Software Directive as applied in the SAS v WPL case. Functional behaviour of computer programs is not part of the literary expression for which copyright protection is available. The SAS programming language should be treated as among the ideas in the SAS program under the Software Directive because it is needed as a tool with which to build other programs (that is, the scripts written by SAS users), rather than being part of the SAS program's expression. While we think the UK High Court should have been somewhat more circumspect in its interpretation of the Software Directive about protection of interfaces, we agree that the data formats in the SAS v WPL case were functioning as interfaces indispensable to achieving interoperability. The Directive permits the development not merely of complementary programs that interoperate with a platform, but also of competing programs that can serve as drop-in replacements for a software platform. Our article thus offered considerable support for the UK High Court's ruling and for Advocate General Bot's Opinion advising the CJEU about the proper interpretation of the Directive.

One notable thing about A.G. Bot's Opinion is that it does not recognise the interconnectedness of the legal issues presented in the SAS v WPL case. This is unquestionably because A.G. Bot was asked to address a specific set of questions identified by the UK High Court. But the issues are deeply interconnected in the SAS v WPL case: WPL could only hope to attract customers to its alternative platform if it emulated the functionality of the SAS platform so that the scripts that customers had constructed would interoperate successfully on WPL's platform. It was also essential for WPL to conform to the data format rules and to the syntax of the SAS language in which scripts were constructed if the scripts were to interoperate on WPL's platform. The behaviour, language and data format issues were not, in the context of this case, three separate legal issues; they were merely three strands of the very kind of interoperability issue that the Software Directive was intended to address.

To an American commentator, it seems quite strange to ask an appellate court such as the CJEU to interpret copyright law—in this case, the Software Directive—in an abstract manner, instead of as applied to the facts of this case. US courts would never render the kind of advisory opinion that CJEU has been asked to produce.

One advantage of the American approach is that US judges pay very close attention to the facts of each case and generally refrain from opining about interpretive issues not presented by the case before them. The Altai and Sega decisions, for instance, were grounded in very specific facts about the necessity to reuse interface information in order to bring into the market new non-infringing products that would compete with existing programs. US courts didn't have to say that computer program interfaces were categorically unprotected by copyright law; they just had to consider reasons why interfaces should not be protected in these specific cases. As later courts found the reasoning in Altai and Sega persuasive, the US common law approach produced the now prevailing consensus that interfaces essential to interoperability are unprotected by copyright law (although they are sometimes patented).
Another advantage of the American approach is that US courts have been able to fashion a number of common law doctrines that provide flexibility in interpreting the often open-ended provisions of copyright law. In the United States, for example, courts could decide that a particular computer program interface was unprotectable because it lacked originality owing to the arbitrariness of interface designs or because the interface consisted of unprotectable facts. US judges could also invoke fair use or the merger of idea and expression doctrine when a computer program interface constrained the design choices of subsequent programmers. To the extent that an interface has become a standard to which other programs needed to conform, they could also apply the scenes a faire doctrine to deny protection. Or, as in Sega, US courts could decide that insofar as interfaces were essential to interoperability, they should be regarded as unprotectable methods of operation under 17 USC §102(b).

European courts have fewer doctrinal tools with which to analyse interoperability issues posed in cases such as SAS v WPL. Perhaps, though, the European tool kit will be enhanced if the CJEU decides that computer program interfaces may be methods of operation under the Software Directive that should be treated as among the programs' ideas.

**Conclusion**

The past and present of interoperability in the European Union and United States has been and is bright. Legislators and courts have recognised the benefits likely to flow from legal rules that enable new entrants to develop complementary and competing products that interoperate with existing programs. The future of interoperability in the European Union will be deeply affected by the CJEU's interpretation of the Software Directive in the SAS v WPL case. Although this directive does not categorically exclude interfaces from the scope of copyright protection in computer programs, it recognises that interfaces may be unprotectable elements of programs. Hopefully the CJEU will perceive the deeply pro-interoperability policies embedded in the Software Directive and offer guidance to courts in conceptualising how copyright law should apply to various kinds of interfaces. A pro-interoperability outcome in SAS v WPL would bring the European Union into harmony with the legal rules affecting interfaces essential to interoperability in the United States. Competition and ongoing innovation, as well as greater consumer choice, will be fostered if the pro-interoperability policies underlying the Software Directive are recognised by the CJEU.