Does Copyright Protection Under the EU Software Directive Extend to Computer Program Behaviour, Languages and Interfaces?

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

Thomas Vinje
Partner, Clifford Chance LLP, Brussels

William Cornish
Emeritus Herchel Smith Professor of Intellectual Property Law, Cambridge University

Copyright; EU law; Infringement; References to European Court; Software

This article argues that competition and innovation in the software industry in the European Union will be seriously undermined if the Court of Justice of the European Union in SAS Institute, Inc. v World Programming Ltd holds that copyright protection for computer programs extends to the functional behaviour of computer programs, to programming languages and to data formats and data interfaces essential for achieving interoperability. This article explains why the text and legislative history of the EU Software Directive, in line with international treaty provisions, should be understood as providing protection for the literary aspects of programs, but not to functionality, languages and data interfaces. Copyright has an important, but limited, role to play in protecting program innovations, especially in view of the increased availability of patents for functional aspects of software.

Twenty years ago, European software copyright law was successfully harmonised by the European Union’s 1991 Software Directive (the Directive). ¹ Now, only a few months after its 20th anniversary, it is suddenly at the centre of attention via a preliminary reference to the Court of Justice of the European Union (CJEU) from the English High Court in SAS Institute, Inc. v World Programming Ltd (WPL).²

In defining the scope of copyright protection for computer programs for the EU, the drafters of the Directive sought to strike a careful balance between, on the one hand, providing appropriate copyright protection to computer programs in order to stimulate investments in new software development, and, on the other hand, enabling second comers to engage in independent development of software capable of fully interoperating with other programs.³ As the recent Microsoft antitrust case demonstrated, if competitors cannot produce software that is interoperable with industry-leading computer programs, consumers will be deprived of innovative competing and/or complementary products.⁴

The SAS v WPL case presents precisely the scenario anticipated by the Directive’s drafters: SAS owns a copyright in a leading statistical analysis program. WPL independently developed a program to compete directly with SAS’s program, with the goal of enabling customers who had constructed computer programs (or “scripts”) in the SAS programming language to run them on WPL’s alternative interoperable platform. WPL had no access to SAS’s source code or other internal design documentation. It simply studied SAS manuals and tested the operations of the SAS program to understand the methods of operation SAS had utilised so that WPL could develop a program to emulate the functionality of the SAS platform. Despite conceding that WPL had copied nothing from SAS source code, SAS nonetheless is seeking an interpretation of the Directive from the CJEU that would prevent WPL from providing consumers with an independently developed alternative with which their SAS-language scripts can fully interoperate as if they were using SAS’s functionally equivalent software.⁵ In other words, SAS is seeking to prevent WPL from doing precisely what the Directive was designed to allow interoperable developers to do.

¹ Thomas Vinje is also Chairman of Clifford Chance’s Global Antitrust Practice and leads its global technology practice. The authors thank Andriani Ferti, associate in Clifford Chance’s Brussels office, for her generous assistance.
² Directive 91/250 on the legal protection of computer programs (the Directive). The 1991 version of the Software Directive was recently replaced by a codified (i.e. consolidated) version, European Parliament and Directive 2009/24 of April 23, 2009, in which both the sequence of the Recitals and the numbering of some of the Directive’s articles were changed. This article will cite to the 1991 Directive in conformity with the High Court decision analysed herein.
³ SAS Institute Inc. v World Programming Ltd [2016] EWHC 1829 (Ch) (SAS v WPL).
The July 2010 decision in *SAS v WPL* issued by Mr Justice Arnold of the English High Court concluded that WPL had not infringed the SAS copyright, because the only substantial similarities between WPL's and SAS's programs lay in their functional behaviour, their use of the same programming language and their ability to process the same data formats, none of which was within the scope of protection that copyright law provides to software developers under the Directive.\(^8\) However, Arnold J. decided to refer a set of questions to the CJEU about the correctness of his interpretation of the Directive on these issues.\(^7\)

In our opinion, Arnold J.'s interpretation of the Directive is sound and should be affirmed by the CJEU. The opinion of Advocate General Bot (A.G. Bot), delivered on November 29, 2011, provides substantial support for the English High Court's interpretation of the Directive as excluding the functional behaviour of programs and programming languages from the scope of copyright protection.\(^7\) While A.G. Bot seems ultimately to support an interpretation of the Directive under which implementation of program interfaces (including the data formats at issue in *SAS v WPL*) should not give rise to copyright liability when this is essential to achieving interoperability,\(^9\) his opinion does not address the question presented by the UK High Court and is more confusing than clarifying on interface issues.

The first part of this article explains why we believe Arnold J. and A.G. Bot correctly concluded that the functional behaviour of computer programs lies outside the scope of software copyright under the Directive. The second part addresses Arnold J.'s conclusion, with which A.G. Bot substantially agrees, that programming languages like SAS's are excluded from copyright protection under the Directive. The third part considers the High Court's conclusions that re-implementing SAS's data formats in WPL's alternative platform falls outside the scope of protection. The fourth part reflects on the role of copyrights and patents in providing legal protection for program innovations.

**Copyright law does not protect functional behaviour of computer programs**

The Directive could not be clearer in its conception of computer programs as copyright subject-matter. Member States are required under this Directive, as well as under the Agreement on Trade-Related Intellectual Property Rights (TRIPS) and the WIPO Copyright Treaty (WCT), to protect computer programs “as literary works within the meaning of the Berne Convention for the Protection of Literary and Artistic Works”\(^10\).

Source code is the clearest manifestation of the literary character of computer programs. Programs typically are written in human-readable source code form in a particular computer programming language. Source code sets forth the set of detailed instructions for performing specified functions in the microprocessor of a computer. When these instructions are transformed into machine-executable form (“object code”), the specified functions can then be carried out by the computer hardware and software platforms for which they were designed. Object code consists of a set of electronic signals representing high and low voltages that instantiate source code instructions. Object code can be transformed back into a printed or viewable textual form by making a “dump” of the program code from a computer’s memory or by decompilation. Object code is as much a “literary work” under the Directive as source code.\(^11\)

The scope of copyright protection for computer programs under the Directive extends beyond source and object code. Translations of programs from one programming language to another and other adaptations, arrangements, or alterations to programs are unquestionably within the scope of protection that the Directive provides to them.\(^12\) Detailed preparatory design materials, such as flowcharts, are manifestations of program design that copyright law protects in the European Union.\(^13\)

Under the Directive, it will thus generally infringe a software copyright for a second comer to write a program using another programmer's preparatory design documents. The decision to include preparatory design materials within the Directive's conception of computer programs indicates the drafters' intent that the detailed structure, sequence, and organisation of the internal design

---

8. *SAS v WPL* [2010] EWHC 1829 (Ch) at [212]-[217], [247]-[248], [255].
9. *SAS v WPL* [2010] EWHC 1829 (Ch) at [332]. Arnold J. referred a total of nine questions to the CJEU, including whether WPL had infringed copyright by developing the alternative platform at issue in violation of a license restriction limiting use of purchased SAS software to individuals for non-production purposes. Initially, this was the only claim in the *SAS v WPL* case; see [277]. Arnold J. ruled that under art.9(1) of the Directive this restriction was unenforceable and hence WPL's development of software in breach of this licence term was not an infringement: [293]. While we do not discuss this aspect of *SAS v WPL* in this article, we concur in this aspect of Arnold J.'s analysis of the Directive as well.
10. *SAS Institute, Inc v World Programming Ltd* (C-406/10) Opinion of A.G. Bot, November 29, 2011 at [57]-[64], [69]-[75].
11. Directive art.1; TRIPS art.10(1); WCT art.4.
12. Directive art.1(2) (indicating that copyright protection is available to the original expression "in any form of a computer program").
13. Directive art.4(b). Error correction is permissible, however: art.5(1).
14. Directive art.1(1)
of program writings be protected by copyright. This is akin to court rulings protecting the detailed structure of conventional literary works, such as novels.

In *SAS v WPL*, Arnold J. accepted that source and object code, as well as detailed structure of program code and preparatory design materials, were protectable expression of programs. He regarded the analogy between the internal structure of programs and the plots of novels to be persuasive insofar as these structures reflect the personal expressive choices of an author of the texts being composed. However, he considered the functionality of programs (that is, their behaviour) and the plots of novels to be non-analogous, as had Mr Justice Pumfrey in *Navigaire Inc v easyJet Airline Co*.

Navigaire had developed airline reservations software, called OpenRes, of which easyJet had been a licensee. Subsequently, easyJet commissioned another firm to develop the eRes program to perform the same functions as OpenRes, and to be, in fact, a drop-in replacement for OpenRes, that is, a program that would accept the same inputs as OpenRes and produce identical results. Pumfrey J. ruled against Navigaire’s claim that easyJet infringed, asserting that:

“[I]t is not possible to infringe copyright that subsists ... in source code ... by observing the behaviour of the final program and constructing another program to do the same thing.”

Pumfrey J. offered this analogy: suppose a talented chef devised a tasty new pudding that another chef wanted to emulate. If the second chef discovered how to make a pudding that tasted the same without accessing the first chef’s recipe, the second chef’s recipe would not infringe the first chef’s copyright. Nor should it infringe copyright, reasoned Pumfrey J., for a firm without access to another firm’s source code or plans for the architecture of a program to develop a functionally equivalent program such as eRes.

Arnold J. quoted at length from the *Navigaire* decision, with which he agreed. He also considered it:

“[R]emarkable that a claim for infringement of copyright can be advanced in which, not only does the claimant not suggest that the defendant ever saw the copyright work, but also the claimant does not consider it necessary for either its own expert witness or the court to see the copyright work.”

Arnold J. also relied on provisions of the TRIPS Agreement and the WCT in support of his conclusion. Both treaties reflect an international consensus that copyright protection for computer programs should not extend to “ideas, procedures, methods of operation, or mathematical concepts.” Arnold J. believed that this provision was added to both treaties out of concern that otherwise software developers might try to monopolise methods of operation and mathematical concepts through copyright law.

Although the Directive refers only to “ideas and principles” as being excluded from the scope of copyright protection, Arnold J. believed that UK copyright law, as well as the Directive, should be construed as consistent with international treaty obligations, and hence, that methods of operation and mathematical concepts embodied in the SAS statistical analysis programs should not be protectable by copyright.

Arnold J. accepted that designing the behaviour of the SAS programs required creative effort, but “[s]kill, judgment, and labour in devising ideas, procedures, methods of operation, and mathematical concepts [of programs] is not protected by copyright in a literary work.” Only “the skill, judgment, and labour in devising the form of expression” in software literary works are pertinent to copyright protection. The behaviour of SAS’s program constituted, in Arnold J.’s view, methods of operation of mathematical concepts, which were unprotectable under UK law.

Further support for this conclusion can be found in the legislative history of the Directive which regarded “the main advantage” of using copyright to protect computer programs as lying in the:

---

13 Cramona and Hart, *Legal Protection of Computer Programs in Europe (1991)*, p.36. See also *SAS v WPL* [2010] EWCH 1829 (Ch) at [184]-[185] (discussing such conventional cases).
14 *SAS v WPL* [2010] EWCH 1829 (Ch) at [232].
15 *SAS v WPL* [2010] EWCH 1829 (Ch) at [234]-[235].
16 *SAS v WPL* [2010] EWCH 1829 (Ch) at [234]-[235].
18 *SAS v WPL* [2010] EWCH 1829 (Ch) at [174]-[176] (discussing Navigaire).
19 *Navigaire* [2004] EWCH 1725 (Ch); [2005] E.C.C. 30 at [86].
21 Navigaire [2004] EWCH 1725 (Ch); [2005] E.C.C. 30 at [94], [113], [129]. Pumfrey J. also observed that “[t]o emulate the action of a piece of software by the writing of other software that has no internal similarities to the first but is deliberately designed to ‘look’ the same and achieve the same results is far from uncommon” at [5].
22 *SAS v WPL* [2010] EWCH 1829 (Ch) at [191].
23 *SAS v WPL* [2010] EWCH 1829 (Ch) at [50].
24 TRIPS art.9(2); WCT art.2.
25 *SAS v WPL* [2010] EWCH 1829 (Ch) at [204] (citing authorities).
26 Directive art.1(2).
27 *SAS v WPL* [2010] EWCH 1829 (Ch) at [199]-[208]; Directive, Recital 15 (Directive should be construed in accord with international treaties).
28 *SAS v WPL* [2010] EWCH 1829 (Ch) at [207].
29 *SAS v WPL* [2010] EWCH 1829 (Ch) at [207].
30 *SAS v WPL* [2010] EWCH 1829 (Ch) at [207].
"fact that the protection covers only the individual
expression of the work and gives thus sufficient
flexibility to permit other authors to create similar
or even identical programs provided that they abstain
from copying." 33

Arnold J.’s ruling against copyright protection for
functional behaviour is consistent with the Directive’s
careful treatment of reverse engineering. The Directive
is very permissive when it comes to “black-box” reverse
engineering, which involves running the software under
various conditions to observe its operations and
understand “the functioning of the program” and to infer
what ideas and principles might be embodied in the
program code that could bring about this functionality. 34
The Directive places no restriction on re-uses of program
functionality or ideas and principles discerned through
black-box analysis. It is highly significant that the
black-box testing privilege cannot be overridden by
contract. 35 This rule was evidently adopted so that it would
be lawful for potential competitors to buy copies of
others’ programs in the marketplace to study them more
closely in the course of developing similar products. 36

The Directive’s permissive approach to black-box
testing contrasts with its more restrictive approach
towards another form of reverse engineering, namely,
decompilation of program object code (i.e. unlicensed
efforts to recreate a facsimile of a program’s source code
through a reverse compilation process). Decompilation
is permitted solely when it is the only way to obtain the
information necessary to achieve interoperability with
other programs. 37

Considered together, arts 5(3) and 6(1) embody a
simple rule: reverse engineering to study functionality is
fine, but reverse engineering to study program code,
internal structure, and other expressive aspects of the
literary character of programs is forbidden, except when
indispensable to interoperability.

In substantial accord with Arnold J.’s opinion and our
observations about the Directive’s intentions as to
program functionality, A.G. Bot’s Opinion casts doubt
on the proposition that the functional behaviour of
computer programs is protectable by copyright law. 38 In
his view, “the functionality, or indeed the combination
of several functionalities, [is] comparable to an idea and
cannot therefore be protected, as such, by copyright law”. 39
To hold otherwise “would amount to making it
possible to monopolise ideas, to the detriment of
technological progress and industrial development”. 40
The level of “skill, judgment and labour expended in devising
the functionality of a computer program” is, in his view,
not germane to the scope of copyright in computer
programs. 41

The exclusion of program functionality from copyright
is consistent with a standard computer science conception
about the nature of computer programs. Computer
scientists consider programs to be “machines that just
happen to be constructed in text”. 42 Computer programs
drive the behaviour of a wide array of machines today—microwave ovens, fuel injection systems in cars,
stock market trading, air traffic control systems, nuclear
power stations, just to name a few. While copyright is
important to protecting program code from unlicensed
copying, the functionality of these software-driven
machines is not protectable by copyright law, but only
(if at all) by patents. 43

If the CJEU interprets the Directive in SAS v WPL as
not extending to program functionality, this would make
EU copyright law more consistent with software copyright
rulings in the United States. In two leading cases, US
appellate courts have rejected claims of copyright
infringement predicated on copying of an existing
program’s functionality or behaviour. In Computer
Associates Int’l, Inc v Altai, Inc, the Second Circuit Court
of Appeals rejected the plaintiff’s infringement claim
because similarities between its and Altai’s scheduling
programs were attributable to their performing the same
set of services for IBM operating system programs. 44
In Lotus Development Corp v Borland Int’l, Inc, the First
Circuit Court of Appeals ruled that Borland had not
infringed copyright by providing an emulation mode that
allowed users who had constructed macros (mini-programs for commonly executed sequences of
functions) in a language devised by Lotus to continue to
eexecute their macros when they switched to Borland’s
spreadsheet program. 45 Borland had unquestionably copied
methods of operation from the Lotus program, but these
methods were outside the scope of protection that

34 Directive art.5(3).
35 Directive art.9(1).
36 A software copyright owner would be unlikely to sue customers for their ordinary uses of the software. But it might want to restrict black-box reverse engineering by potential competitors, as SAS has tried to do as to WPL. See SAS v WPL [2010] EWIIC 1829 (Ch) at [267]-[293], Arnold J. ruled that the licence restriction that SAS asserted against WPL was unenforceable under art.9(1) of the Directive: at [311]-[133].
37 Directive art.6(1).
41 "SAS Institute Opinion of A.G. Bot, November 29, 2011 at [64].
43 Huet and Ginsburg, "Computer Programs in Europe" (1992) 30 Columbia Journal of Transnational Law 327, 336. A substantial number of patents have issued for software functional properties since the Directive was adopted. See below, pp.165-166.
45 Lotus Development Corp v Borland Int’l, Inc 49 F. 3d 807 (9th Cir. 1995).
copyright law provides to programs. This is comparable to the reimplementation of the statistical methods of operation of the SAS program in independently written code prepared by WPL.

**Programming languages such as SAS’s are unprotectable by copyright law**

The Directive was unquestionably intended to protect “only the expression of a computer program”, not its ideas or principles; it recognises that “to the extent that logic, algorithms, and programming languages comprise ideas and principles”, these aspects of programs are unprotected by copyright law.

At first blush, this presents something of a puzzle: the Directive contemplates that programming languages may comprise unprotectable ideas and principles; yet it signals that this will not always be so through the qualifying phrase “to the extent that …”. SAS’s counsel seized upon this qualification to argue that the SAS language should be treated as copyright-protected expression for two interconnected reasons: first, that the Directive did not categorically exclude programming languages from the scope of copyright protection, and secondly, that programming languages were only outside copyright’s protection if they were abstract in the same sense as ideas.

SAS’s counsel is mistaken. Algorithms, logic, and programming languages—all of which are subject to the same qualification—comprise, by their very nature, specific and detailed ideas, not highly abstract ones. These components of software should be excluded from copyright protection when their free use is needed as tools through which computer programs are constructed. We think that the Directive’s qualification sought to avoid Jesuitical quarrels over categories (is the challenged element a “programming language” or not?). It aims instead to encourage courts to engage in a careful examination, first, of respects in which the challenged subject-matter in a software copyright case plausibly fits within standard computer science definitions of “programming language”, and secondly, whether under the circumstances of a particular case, the policy reasons for treating a putative language as outside the scope of copyright protection outweigh the policy reasons that would favour treating it as protectable expression. If the former outweigh the latter because free use of a programming language is needed as a building block for constructing new programs, then the programming language should be among the ideas and principles that copyright does not protect in programs.

In *SAS v WPL*, it was relevant that SAS itself, outside of the litigation with WPL, had consistently spoken of SAS scripts as being written in “the SAS programming language”. Arnold J. also considered expert testimony and reports proffered by both litigants on the programming language issue. Although SAS’s expert had prepared a chart setting forth different types of programming languages in an attempt to distinguish the SAS language from other types of programming languages; Arnold J. found WPL’s expert more persuasive. WPL’s expert explained to the court that the SAS language conformed to the standard definition of programming language in an IEEE glossary of computer terms (“A language used to express computer programs”) because users of SAS used the SAS language to construct scripts, which are computer programs.

Arnold J. could have done more, however, to explain why the SAS programming language should be considered the kind of language that the Directive considers unprotectable. He mainly relied on Pumfrey J.’s assertion in *Navitaire* that the Directive was intended “to keep the language free for use”. A.G. Bot’s analysis of the programming language issue nicely augments this key point by noting that a programming language “devises the specific methods to be used and facilitates thinking necessary to write and formalize computer source programs”. This led A.G. Bot to conclude that a programming language like SAS’s is “a functional element which allows instructions to be given to the computer” and hence “the means which permits expression to be given, not the expression itself”.

It is worth noting that programming languages, like languages more generally, consist of three elements: a vocabulary, a set of semantics and a syntax for utilizing the language to compose a text in the language. Programming languages are abstract in the sense that they provide the building blocks with which to compose such a text, as the SAS language unquestionably did. Consequently, the SAS language should be considered among the ideas and principles in the SAS programs that copyright does not reach.

Further support for the conclusion that the SAS language should be treated as ideas and principles derives from the substantial investments that SAS users have made over the years in composing many thousands of

---

46 *Lotus v Borland* 49 F. 3d 807, 816 (9th Cir. 1995).
48 Directive, Recital 15.
49 *SAS v WPL* [2010] EWHC 1829 (Ch) at [210]-[211].
50 *SAS v WPL* [2010] EWHC 1829 (Ch) at [47]-[48]. SAS’s own materials pointed to similarities between the SAS language and other programming languages: at [48].
51 *SAS v WPL* [2010] EWHC 1829 (Ch) at [51]. SAS’s expert distinguished between general purpose computer programming languages (such as C or FORTRAN) and domain specific languages, which he thought included program-specific command languages and languages such as SAS’s. WPL’s expert opined that there was no scientific basis for this ontology proffered by SAS’s expert: at [53].
52 *SAS v WPL* [2010] EWHC 1829 (Ch) at [49]-[55], [212]-[218].
53 *SAS v WPL* [2010] EWHC 1829 (Ch) at [35]-[54].
54 *SAS v WPL* [2010] EWHC 1829 (Ch) at [176]-[179], [209]. Pumfrey J., in turn, relied on a UK copyright treatise that regarded programming languages as unprotectable by copyright law. *Navitaire* [2004] EWHC 1725 (Ch); [2005] E.C.C. 30 at [85].
57 *SAS v WPL* [2010] EWHC 1829 (Ch) at [2].
scripts in the SAS language to run specialised mathematical operations on the SAS platform. Until WPL came along, SAS language scripts could only be executed on the SAS platform.

This meant that "a [SAS] customer who wanted to change over to another supplier's software would be faced with rewriting its existing applications programs in a different language". Users were thus "locked in" to the SAS platform. They might thus remain SAS customers not so much out of satisfaction with overall superiority of SAS's software or the terms on which it was offered, but because they had made sunk-cost investments in writing scripts. For many users, the expense of rewriting scripts was an impediment to switching to a newer and possibly better platform. WPL reduced the switching costs for users by providing an alternative platform on which SAS scripts could be executed, and it did so without access to or copying from the literary expressions of the SAS programs. It was essential to WPL's strategy of attracting dissatisfied SAS customers that its platform could accept the same inputs (scripts written by users in the SAS language) and produce the same outputs (the same computational results as if the scripts were performed on the SAS platform).

These same policy considerations influenced US court decisions rejecting claims that programming languages are protectable "expression". Affirming Arnold J.'s ruling on programming languages in the SAS v WPL case would make EU software copyright law consistent with US. precedents.

To what extent does copyright law protect interfaces?

Under the Directive, copyright protection does not extend to computer program interfaces insofar as they define the rules and methods for exchanging information to which new programs must conform if the new programs are to achieve interoperability with existing programs. However, the Directive does not per se exclude all program interfaces from protection as ideas and principles. Thus, while we agree with Arno d J.'s ruling that WPL's ability to process SAS data formats is not infringement, Arnold J. arguably overstated the extent to which interfaces are excluded from copyright protection under the Directive. A.G. Bot, on the other hand, took an unduly narrow view of the Directive's treatment of interfaces and under-appreciated the extent to which the Directive was meant to foster interoperability. We offer in this part a middle-ground perspective under which WPL should prevail without posing risks of under- or over-protection of interfaces.

Several aspects of the Directive reflect a strong legislative policy in favour of facilitating interoperability, which consequently calls for careful analysis of infringement claims regarding interfaces, because interfaces are the elements of computer programs through which programs are able to interoperate with other programs, hardware and data.

First, the specific reference in art.1(2) to interfaces not only reiterates the general principle of copyright law according to which only expression and not ideas and principles underlying interfaces are eligible for protection, but in addition serves to emphasise that many interface elements would qualify as unprotected ideas rather than protected expression.

Secondly, several Recitals of the Directive emphasise the unique nature of computer programs and the imperative of interoperability to which they are subject. Unlike other copyrighted works, computer programs must "communicate and work together with other components of a computer system" which means 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 ... with which they are intended to function."

The Recitals observe that "the parts of the program which provide for such interconnection and interaction are interfaces" and that "this functional interconnection and interaction is generally known as interoperability".

These Recitals unquestionably reflect a policy in favour of interoperability and a legislative directive to courts to be careful in delineating the appropriate scope of protection in any case where interoperability is at stake.

Thirdly, the very existence of art.6, which permits access via decompilation of a program's code to study its internal structure and other expressive aspects of program text when this process is indispensable to achieving interoperability, further reflects the pro-interoperability policy underlying the Directive. It would make no sense to establish a right to access program internals, and then to condemn the reuse of interface information discerned in this process as infringing when it is necessary to achieve interoperability.
Given these provisions, the Directive should be understood as encouraging courts to carefully examine whether the interface elements at issue in a particular case constitute uncopyrighted ideas or methods of operation, as will often be the case, or might instead constitute protectable expression.

Arnold J. seemingly believed that as soon as a court has concluded that a program element constitutes an “interface”, copyright protection is unavailable.66 This fails to take into account two important facts: first, there are different types of interfaces, some of which might be more susceptible to copyright protection than others, and secondly, there exists a vital distinction between the implementation of an interface in program code (which, when original, will generally constitute protectable expression) and the rules and methods underlying an interface, which govern program behaviour and are outside the scope of copyright for reasons explained in the first part of the article.

Among the basic types of interfaces are: (1) data formats, such as those at issue in SAS v WPL; (2) communication protocols; (3) application programming interfaces (APIs); and (4) user interfaces.76 The latter enable interaction between a computer system and its users, and typically do not involve interoperability issues addressed in the Directive.

A.G. Bot correctly notes that data formats such as those at issue in SAS v WPL:

“[M]ay be regarded as black forms which are to be filled with the customer’s data … and which contain specific locations in which particular information must be written in order for the system to read and write the file correctly.”77

Because these data formats allow users to set up a program to exchange information and mutually to use the information which has been exchanged”, they satisfy the Directive’s definition of interfaces, and the interaction they enable conforms to the Directive’s definition of interoperability.72

That WPL utilises these data formats to facilitate interoperability with user scripts, not the SAS program, does not undermine its defence. The legislative history of the Directive demonstrates that it was intended to enable both types of interoperability. Several early drafts of the Directive provided that decomposition would be permissible solely to achieve interoperability with “the original program” (i.e. with the decomposed program).

However, after considerable debate over this provision, art.6 now permits decomposition to achieve interoperability with “other programs” (i.e. not necessarily or not solely with the decomposed program, but also (or instead) with other programs).75 As explicitly noted by the European Commission in its communication to the European Parliament,76 decomposition is permitted only for purposes of producing complementary products, but also “competing” products. This means that art.6 permits decomposition for purposes of ascertaining the interfaces of an existing program (e.g. SAS) so that those interfaces can be implemented in a new, independently created program (e.g. WPS) that enables the new program to interoperate with other programs (e.g. user scripts written in the SAS programming language) in the way the existing program does.75

Given that art.6 permits reverse engineering (decomposition) to ascertain interfaces necessary to create programs that interoperate with other programs in a manner that enables “drop-in replacement” for the program being decomposed (i.e. a manner that enables them to interoperate in a manner that competes with the first program), the Directive should be understood as also enabling interoperability and re-using data formats (even when the second comer has developed a competing platform) where there has been no decomposition.

Data formats, such as those at issue in SAS v WPL, are a type of interface that prescribes the arbitrary, abstract rules and methods according to which data must be formatted in order to create a computer program to process that data. SAS users may format the data as prescribed by SAS. In order for WPL’s program to process that data, it must comply with the rules established by SAS. To accomplish this, WPL need not copy anything from the SAS source code or any arguably expressive elements of the SAS program; indeed, it had no access to these aspects of SAS’s program.76

There is nothing expressive about data formats insofar as they arbitrarily prescribe the method according to which data must be formatted (e.g. the order and length of the data fields) in order for a program to process the data.77 A.G. Bot was mistaken when he said that the SAS program contains data formats.78 The more accurate formulation is that the SAS program is able to process user data when it and scripts written in the SAS language conform to the prescribed format. WPL had to conform

66 SAS v WPL [2010] EWHC 1829 (Ch) at [226].
73 SAS v WPL [2010] EWHC 1829 (Ch) at [3].
to the same rules so that its program could \textit{behave} in the same way as the SAS program and \textit{process} the same user data.

Arnold J.'s recognition of this point\textsuperscript{78} affected how he formulated his question to the CJEU. He did not ask whether data formats were protectable, but rather whether it is "an infringement of the copyright in the First Program for the Second Program to be written so as to read and write to data files in the same format".\textsuperscript{79} Unfortunately, A.G. Bot did not address this question.

Although we agree with A.G. Bot's conclusion that the Directive does not per se exclude interfaces from copyright protection and that "if the expression of the interface constitutes a substantial part of the expression of the computer program ... it is eligible for copyright protection",\textsuperscript{80} most of A.G. Bot's discussion of the interface issue addresses a question directly contrary to the facts found in \textit{SAS v WPL}. A.G. Bot considers whether WPL "was entitled to perform an act of decompilation in order to achieve interoperability between the SAS System and its WPS System".\textsuperscript{81} Yet the English High Court expressly found that WPL did not decompile SAS code.\textsuperscript{82} WPL discerned what it needed to know in order to achieve interoperability with scripts written in the SAS language by reading about the behaviour of the SAS program in the SAS manuals and by undertaking "black box" analysis of the SAS program.

A.G. Bot's apparent belief that the case involved decompilation seemingly caused him to address a question quite different from that posed by the English High Court. A.G. Bot says that:

\begin{quote}
"[T]he referring court asks whether, in essence, WPL has committed an infringement of copyright by \textit{deciphering} enough of the format of the SAS data files to be able to write a source code, in its own computer program, which reads and writes data files in the same format."\textsuperscript{83}
\end{quote}

The English court did not, however, ask anything about "deciphering" in its question regarding data formats; it asked whether it infringes copyright for a second program "to be written so as to read from and write to data files in the same format".

We hope the CJEU will address the specific question asked by the High Court, and that it will unambiguously state that no infringement occurs when someone writes a program that can read and write to data files in the same format as an existing program because this only replicates the \textit{behaviour}, and not the \textit{expression}, of a computer program.

Such a ruling would make EU law consistent with US copyright law. In the leading US case, \textit{Computer Associates Int'l v Altai, Inc.}, the Second Circuit Court of Appeals ruled that Altai's reimplementation of interfaces discerned from studying CA's program did not infringe copyright because these interfaces constrained Altai's design choices insofar as it wanted to make its scheduling program equally compatible as CA's with IBM operating system programs.\textsuperscript{84} Altai's interoperability defence was not affected by the fact that its program directly competed with CA's program. Moreover, in \textit{Lotus v Borland}, as in \textit{SAS v WPL}, the plaintiff sued to stop a competitor from offering an alternative platform on which users could execute programs constructed in a language devised by Lotus. Borland prevailed in that lawsuit because it had only copied unprotectable methods of operation from the Lotus program, not expression.\textsuperscript{85}

\textbf{The role of copyright in the larger intellectual property framework for protecting computer programs}

The arguments in \textit{SAS v WPL} have rightly turned on the meaning to be given to the Directive, in light of provisions in international treaties and other EU Directives on copyright, which regulate the originality required for a work to attract copyright and the idea/expression dichotomy as it affects the scope of copyright protection. There is, however, a broader comparison which must be considered in deciding whether software copyright should extend to functional behaviour, programming languages and data formats affecting interoperability.\textsuperscript{86} It is fundamentally important to consider the role of copyright in protecting software alongside the protection of inventions provided by the patent system. After all, computer programs are developed to become machine processes by means of textual coding.

In Europe software patenting has had a chequered history, beginning with the exclusion of computer programs "as such" from the realm of patentable subject-matter in the European Patent Convention of 1973.\textsuperscript{87} The equivocal "as such" formula led some to argue that patents for programs should be available or the same terms as for other technologies because programs determine the functioning of computers, while others have sought to limit the availability of patents for software innovations through special legal rules. In the last decade, the European Commission tried to settle the difference through a new Directive; but this effort foundered owing largely to opposition from advocates of "free software".\textsuperscript{88}

\textsuperscript{78} \textit{SAS v WPL} [2010] EWHC 1829 (Ch) at [34].
\textsuperscript{79} \textit{SAS Institute Opinion of A.G. Bot, November 29, 2011 at [35].}
\textsuperscript{80} \textit{SAS Institute Opinion of A.G. Bot, November 29, 2011 at [82].}
\textsuperscript{81} \textit{SAS Institute Opinion of A.G. Bot, November 29, 2011 at [83].}
\textsuperscript{82} \textit{SAS Institute Opinion of A.G. Bot, November 29, 2011 at [69].}
\textsuperscript{83} \textit{SAS Institute Opinion of A.G. Bot, November 29, 2011 at [77].}
\textsuperscript{84} \textit{Altai 982 F. 2d 693, 714–715 (2d Cir. 1992).}
\textsuperscript{85} See \textit{Lotus v Borland} 49 F. 3d 897 (1st Cir. 1995), aff'd 516 U.S. 233 (1996).
\textsuperscript{86} See Directive art.9(1) (recognising that patents may also protect programs).
\textsuperscript{87} This occurred largely at the behest of leading firms in the nascent computer industry at the time. For the way in which the issue developed through case law, see, e.g., W.R. Cornish, D. Llewelyn and T. Aplin, \textit{Intellectual Property}, 7th edn (2010), paras 20.26 et seq.
\textsuperscript{88} Cornish, Llewelyn and Aplin, \textit{Intellectual Property} (2010), para 20.27.
The consequence of the failure of this effort, not apparently foreseen by these objectors, has been a series of decisions by Technical Boards of the EPO which have had the effect of neutering the exclusion of computer programs "as such" in the European system wherever the program is claimed for use in a computer. Instead, the grant of a software patent is now determined primarily by the requirements that the claimed invention be novel and non-obvious taking account only of its aspects which produce a technical effect.98

The English Court of Appeal condemned this approach as being much rougher treatment of Convention language than amounts merely to interpretation.99 Patent jurisdictions of the EPC contracting states are, however, likely to follow the path laid out by the EPO tribunals. This approach has the merit of testing the inventive character of software by reference to the same criteria as other technologies, in line with the TRIPS Agreement.100

Hard experience over two centuries has demonstrated the dangers of allowing ideas that do not amount to inventions to be the subject of patent monopolies. One consequence is that, in leading patent systems today, inventions stand to be tested first during the application stage and again, if, after grant a third party challenges its validity.101 This legal structure amounts to a costly prophylactic, but it is accepted as necessary in order to curb those who assert a patent monopoly without being able to justify their claim to it. Patents are thus a cautiously accorded exception to the general freedom to appropriate innovations by reverse engineering what is available on the market or through developing ideas revealed in the relevant literature.

Should software copyright cover the same technical ground as patents by protecting functional behaviour without any copying of substantial portions of code? To extend software IP so far would provide a shortcut to gaining exclusivity over technical alternatives that behave in the same or a similar way to software already available. The exclusion of competition would necessarily be far wider than through software patents. The copyright would come into effect simply upon the writing of the program. It would be infringed when a third party produced a program in its own code which functioned in the same way, whether or not it was written to interoperate with the first product. In truth, rights over functional effects of machines should never be accorded on an unregistered basis.102 This is especially so, where the legal tests of validity are confined to the originality of the program text which expresses the writer’s personal creativity and which is infringed by copying. Providing protection of functional effects of machines via copyright would lead to a thicket of IP barriers which could seriously deprive markets of the general advantages of competition.

Conclusion

Competition and innovation in the software industry in the European Union will be seriously undermined if the CJEU rules in SAS v WPL that copyright protection for computer programs extends to the functional behaviour of computer programs, to programming languages, and to elements of programs that serve as interfaces essential for achieving interoperability. Open source programming may be especially hard-hit as this often aims to emulate the functionality of existing programs. This article has explained why the text and legislative history of the Directive, in line with international treaty provisions, should be understood as providing protection for the literary aspects of programs, but not to functionality, languages, and interfaces. Copyright has an important, but limited, role to play in protecting program innovations, especially in view of the increased availability of patents for functional software designs.

99 See, esp., Aerotel v Telesh. Macrossan v Application [2007] R.P.C. 7 CA (Civ Div); Symbian v Comptroller-General [2009] R.P.C. 1 CA (Civ Div); Cornish, Llewelyn and Aplin, Intellectual Property (2010), para.20.32. An attempt to have the EPO’s Enlarged Board of Appeal review the controversy was confounded by its ruling that there was no conflict in the EPO’s case law on which the President of the EPC could base her reference to it; see Patentability of Programs for Computers (C-388).
100 TRIPS art.27(1): patents shall be available “without discrimination as to ... the field of technology ...”.
102 An example of such consequences is also illustrated by the introduction of unregistered design right into UK copyright law, originally by the Design Copyright Act 1988, which had to be significantly curtailed by the Copyright, Designs and Patents Act 1998 Pt III. For the gross consequences of these changes in the law, see Cornish, Llewelyn and Aplin, Intellectual Property (2010), Ch.15.