3

Copyright, Contract, and Licensing in Open Source

P McCoy Smith

3.1 Copyright and Software

3.1.1 The history of software and copyright

Although the history of manmade computing devices reaches into antiquity,[1] it was not until the development, beginning in the 1950s, of high-level programing languages for authoring and editing computer instructions,[2] together with existing systems for transcribing and loading those instructions into a computing device,

[1] Jo Marchant, 'Decoding the Antikythera Mechanism, the First Computer' *Smithsonian Magazine* (February 2015) <https://www.smithsonianmag.com/history/decoding-antikythera-mechanism-first-computer-180953979/> accessed 11 November 2019.

[2] FORTRAN (a portmanteau of 'formula translation'), first used in 1954, is generally considered to be the first high-level programing language, and remained the dominant programing language for scientific and mathematical computing well into the late twentieth century. See 'Fortran', Techopedia, <https://www.techopedia.com/definition/24111/fortran> accessed 21 December 2016. In fact, several important benchmarks for measuring and comparing performance in supercomputing use FORTRAN, given its frequent use in highly complex scientific and mathematical calculations. See SPEC CPU®2017 Floating Point <http://www.spec.org/cpu2017/Docs/#benchmarks> accessed 13 April 2022.

P McCoy Smith, *Copyright, Contract, and Licensing in Open Source* In: *Open Source Law, Policy and Practice.*
Edited by: Amanda Brock, Oxford University Press. © P McCoy Smith 2022. DOI: 10.1093/oso/9780198862345.003.0003

that software began to resemble those things—such as literary or other artistic works—for which Intellectual Property (IP) rights had previously been extended. The ability to author, adapt, reproduce, and systematically load programing instructions into a computing device using a combination of a writable medium, and a high-level programing language—understandable to a multitude of human programers and (after compilation) computing machines—first began to cause computer scientists, lawyers, and legislators to contemplate forms of IP protection that might be used for programs created using a combination of these mechanisms.

In the 1960s, the potential for using copyright as a mechanism for securing exclusive rights to computer code started to emerge, and test cases were attempted in the US to establish the application of copyright to software.[3] Thereafter, the Commission on New Technological Uses of Copyrighted Works (CONTU) was established in the US in order to study what, if any, IP protection might be appropriate for 'new' technologies like software.[4]

CONTU's eventual report, issued in 1978, recommended that copyright protection should be available for computer programs composed of 'a set of statements or instructions to be used *directly or indirectly* in a computer in order to bring about a certain result'.[5] This recommendation was not without dissent,[6] as some committee members felt that computer software was primarily or exclusively functional and therefore not an appropriate target for copyright protection, but instead should only be protectable by patent. The US Congress amended the US Copyright Act in 1980 to specify that computer software was within its scope.

The change made in US copyright law to reflect the recognition that software fell within its ambit were later included in law in the UK[7] and the EU[8] to give similar recognition.

Modern computing devices are typically configured using, and have data input in, information in a format often referred to as 'binaries' or 'executables',[9] instantiated as a lengthy string of '1' and '0' values. In the early days of the application of copyright to software, there remained the question of whether that

[3] See George D Carey, 'Copyright Registration and Computer Programs' (1964) 11 *Bulletin of the Copyright Society of the United States of America* 362, at 363; General Atomic Division of General Dynamics, 'Gaze-2, A One-Dimensional, Multigroup, Neutron Diffusion Theory Code for the IBM-7090', US Copyright Registration No. A607663 (registered 1 January 1963) (a FORTRAN program, considered to be the first registered copyright on software in the US).

[4] National Commission on New Technology Uses of Copyrighted Works, 'Final Report' (31 July 1978) (CONTU Report).

[5] CONTU Report, see note 4, Ch. 3.

[6] CONTU Report, see note 4, Ch. 3, Concurrence of Commissioner Nimmer (arguing that only certain types of computer programs should be afforded copyright protection) *and* Dissent of Commissioner Hersey (arguing that copyright protections should not be afforded to computer programs at all).

[7] See UK Statutory Instruments 1992 No. 3233 The Copyright (Computer Programs) Regulations.

[8] See Council Directive on the Legal Protection of Computer Programs, 34 OJ EUR. Comm. Mr. (No. L 122) 42 (1991).

[9] See *Encyclopaedia Britannica*, 'Binary Code' (19 January 2020) <https://www.britannica.com/technology/binary-code> accessed 10 February 2020; *PC Magazine Encyclopedia*, 'Executable Code' <https://www.pcmag.com/encyclopedia/term/executable-code> accessed 10 February 2020.

form of software was eligible for copyright protection, given the seemingly pure functionality of the way that code was used and its general incomprehensibility in that format to even skilled programers. A similar issue had confronted courts in both the US and the UK in the early twentieth century, in cases involving 'piano rolls'.[10] In both countries, the courts determined that this format was *not* an infringement of the copyright in the underlying musical composition: '[T]o play an instrument from a sheet of music which appears to the eye is one thing; to play an instrument with a perforated sheet which itself forms part of the mechanism which produces the music is quite another thing',[11] and 'These perforated rolls are parts of a machine which, when duly applied and properly operated in connection with the mechanism to which they are adapted, produce musical tones in harmonious combination. But we cannot think that they are copies within the meaning of the copyright act.'[12] Although these decisions would—if still applicable today—provide a sound basis for denying copyright rights to (or at least, disallowing infringement claims by source code authors against) executable code created using copyrightable source code, subsequent legislative developments reversed the outcome of those decisions.[13] The piano roll decisions, and subsequent legislative changes to reverse them and later to bring computer software within the purview of copyright, were influential—if not dispositive—on the question of whether executable code fell within the protection of copyright when challenges to that proposition were made in the early 1980s.[14] As a result, it is without question that a programer's copyright rights subsist in any instantiation of their authored code—from the form in which the author originally wrote it (in most cases, source code), to any subsequent human or machine translation of it into any intermediate or final format (e.g. object code or executables/binaries), that is understandable by a computing device and upon which it may act, as long as the work otherwise falls within the boundaries of copyright and outside of any exceptions thereto.

[10] A piano roll was a long roll of paper with punched holes which, when fed into a specially adapted piano (called a 'player piano') would convert the information on that roll into played music. See *Encyclopaedia Britannica*, 'Player Piano' (9 September 2019) <https://www.britannica.com/art/player-piano> accessed 10 February 2020.

[11] *Boosey v Whight*, 1900 1 Ch. 122, 81 LTNS 265.

[12] *White-Smith Music Publishing Co. v Apollo Co.*, 209 US 1 (1908).

[13] See Kal Raustiala and Christopher Jon Sprigman, 'Scales of justice: How a terrible Supreme Court decision about player pianos made the cover song what it is today' *Slate* (12 May 2014) <https://slate.com/technology/2014/05/white-smith-music-case-a-terrible-1908-supreme-court-decision-on-player-pianos.html> accessed 13 April 2022; UK Public General Acts 1911 c. 46, Part I, Section 1(2)(d) ('in the case of a literary, dramatic, or musical work, to make any record, perforated roll, … *or other contriavance by means of which the work may be mechanically performed or delivered*'); 17 USC § 102 (2018) ('Copyright protection subsists … in original works of authorship fixed in any tangible medium of expression, now known or later developed, from which they can be perceived, reproduced, or otherwise communicated, either directly *or with the aid of a machine or device*.').

[14] See *Apple Computer, Inc. v Franklin Computer Corp.*, 714 F.2d 1240 (3d Cir. 1983), *cert. dism'd by stip.*, 464 US 1033 (1984).

3.1.2 The author's rights in software copyright

Copyright defines a set of legal rights which are granted—via national law, but in most jurisdictions consistent with the Berne Convention for the Protection of Literary and Artistic Works[15]—to the authors of a work eligible for protection. As discussed earlier, although the question of whether software fell within the copyright protection regime was not resolved at the time that computer software was reaching a state where it began to resemble a work of authorship, legislation, treaties, and to a certain extent national court decisions, have firmly established that copyright protects such works, in any form they take.[16]

Copyright gives exclusive rights to the author to have dominion over certain activities of others when those activities involve the works to which the author has received copyright protection. A recitation of verbs articulates the actions of others over which the copyright holder may either grant permission to do (via licence), or prevent from doing (via enforcement actions) (see Table 3.1).

National laws articulate these rights using slightly different terminology, although doing so generally in conformance with the Berne Convention framework; because many of the Open Source licences in common use were first drafted in the US, they often use the terminology from US copyright law (together with, or as an alternative to, the Berne Convention formulation). For this reason, any particular Open Source licence may have a degree of ambiguity as to whether all relevant copyright rights are conveyed, expressly or by implication; as shown in Table 3.1 above, there is a potential concordance of, *inter alia*, the verbs used in US copyright law with the verbs used in other national law or international treaties.

Open Source licences all express their permissions using at least some of these verbs,[17] and virtually all attach certain conditions (even if fairly minimal and easy to satisfy, as with permissive licences) to those permissions so as to keep the licensed code Open Source, or at least allow users to understand the code is based upon Open Source code, and possibly to seek out and receive the corresponding source upon which the code is based.

[15] Berne Convention for the Protection of Literary and Artistic Works (9 September 1886; as revised through 28 September 1979).

[16] It is notable that the Berne Convention does not specify that software falls within its ambit. See Geoffrey S Kercsmar, 'Computer Software & Copyright Law: The Growth of Intellectual Property Rights in Germany' (1 May 1997) 15(3) *Penn State International Law Review* article 7, at 567. Subsequent treaties, however, have established that Berne Convention compliance does include providing copyright protection to software. See Michael Lehmann, 'TRIPS, the Berne Convention, and Legal Hybrids' (December 1994) 94(8) *Columbia Law Review* 2621, at 2625.

[17] Note that many of the older licences may not use the term 'copyright' when authorising activities. Nevertheless, using verbs that come from national laws, or international treaties, governing copyright, would generally be interpreted to confer copyright rights. The extent to which leaving out some of the copyright verbs from a licence permission should be interpreted as a reservation of that right by the author, or instead the non-recited verbs should be understood by implication also to be granted, is an unresolved issue with some Open Source licences. See, e.g., Andrew Sinclair, 'Licence Profile: BSD' (2015) 1(1) *Journal of Open Law, Technology & Society* 1, at 3.

Table 3.1 Potential Concordance of Copyright Verbs in Various Treaties and National Laws

Berne Convention[a]	UK Copyright Act[b]	EU Copyright Directive[c]	US Copyright Law[d]
Translating	Adapting	Translating	Preparing Derivative Works
Reproducing	Copying	Reproducing	Reproducing
Performing	Performing		Performing
Broadcasting	Showing, Playing, Communicating	Distributing	Distributing, Displaying
Reciting			
Communicating	Issuing copies	Distributing	Distributing
Adapting	Adapting	Adapting, Altering	Preparing Derivative Works
Arranging		Arranging	Preparing Derivative Works

[a] See Berne Convention, see note 15, arts 8, 9, 11, 11*bis*, 11*ter*, 12, 14.

[b] See UK Copyright, Designs and Patents Act 1988, §§ 16–21.

[c] See Directive 2009/24/EC of the European Parliament and of the Council of 23 April 2009 on the legal protection of computer programs (Codified version) (2009).

[d] See 17 USC §106 (2010).

3.1.3 Exceptions to the author's rights in software copyright

3.1.3.1 Non-copyrightability

Although the copyrightability of software, in both source and binary form, has been well-established worldwide since at least 1993,[18] there has always been a tension in providing copyright protection to software given that software is composed of a substantial amount of material that is highly functional.[19] The European Union has made clear in its own software directive that:

> [O]nly the expression of a computer program is protected … ideas and principles which underlie any element of a program, including those which underlie its interfaces, are not protected by copyright under this Directive. In accordance with this principle of copyright, to the extent that logic, algorithms and programming languages comprise ideas and principles, those ideas and principles are not

[18] See Lehmann, 'TRIPS, the Berne Convention, and Legal Hybrids', note 16, at 2625.

[19] *Navitaire Inc. v Easyjet Airline Co* [2004] EWHC 1725 (Ch).

> protected under this Directive. In accordance with the legislation and case-law of the Member States and the international copyright conventions, the expression of those ideas and principles is to be protected by copyright.[20]

The US has a similar concept, the so-called idea/expression dichotomy, established via a long line of court decisions.[21]

Navigating the distinction between that in software which is not copyrightable (and thus available for non-authors to use without adhering to the conditions of any particular licence), and that which is, is not an insubstantial task. The European Court of Justice (ECJ) addressed this issue in 2012 *SAS Institute* case,[22] stating 'the ideas and principles which underlie any element of a computer program, including those which underlie its interfaces, are not protected by copyright'.[23] A similar issue in the US was the subject of a long-standing and significantly publicised legal dispute, eventually resolved by the Supreme Court of the US.[24] That dispute involved an allegation by Oracle that Google copied certain application programing interfaces (APIs), and possibly other information,[25] from Oracle's Java programs without the benefit of a licence from Oracle. Google was successful at the trial court level in arguing that everything they copied was non-copyrightable,[26] while Oracle was successful at the intermediate appeal court level in arguing that some of what was copied was copyrightable.[27] Subsequently, the case was sent to the trial court, where a jury determined that Google's copying fell within the 'fair use' exceptions to US copyright law.[28] That determination was also overturned by the intermediate appeal court.[29] Thus, the US Supreme Court was presented with two questions: (i) were the interfaces copied by Google copyrightable at all, and (ii) if they were copyrightable, was Google's copying nevertheless permissible as 'fair use' under US copyright law?[30]

[20] 'Directive 2009/24/EC of the European Parliament and of the Council of 23 April 2009 on the legal protection of computer programs' Official Journal of the European Union, L 111/17, no. 11 (5 May 2009).

[21] See *Baker v Selden*, 101 US 99 (1879); *Mazer v Stein*, 347 US 201 (1954).

[22] See *SAS Institute Inc. v World Programming Ltd,* ECLI:EU:C:2012:259 (2 May 2012).

[23] *SAS Institute*, see note 26, at 31.

[24] See *Google LLC v Oracle Am., Inc.*, 593 US _ , 141 S. Ct. 1183, Docket No. 18–956, (2021).

[25] Part of the dispute between Google and Oracle, and part of the reason why the trial court and intermediate appeal court rendered differing decisions on copyrightability, relates to whether what Google copied was necessary to interface with Java, or not. *See Oracle America, Inc. v Google LLC*, 886 F.3d 1179 (Fed. Cir. 2018) (analysis of the necessity of 11,500 lines of copied code to allow interoperability).

[26] *Oracle Am., Inc. v Google LLC*, 872 F. Supp. 2d 974 (N.D. Cal. 2012).

[27] *Oracle Am., Inc. v Google LLC*, 750 F.3d 1339 (Fed. Cir. 2014).

[28] David Goldman, 'Jury sides with Google in billion dollar Oracle suit', *CNN Business* (26 May 2016) <https://money.cnn.com/2016/05/26/technology/google-oracle/index.html> accessed 6 April 2021.

[29] *Oracle America, Inc. v Google LLC*, 886 F.3d 1179 at 106 (Fed. Cir. 2018).

[30] *Google LLC v Oracle America, Inc.*, Docket No. 18–956, Petition for a Writ of Certiorari (US 24 January 2019).

Unlike the ECJ in the *SAS Institute* decision, the US Supreme Court in *Google v Oracle* avoided squarely addressing the issue of the copyrightability of software interfaces:

> Given the rapidly changing technological, economic, and business-related circumstances, we believe we should not answer more than is necessary to resolve the parties' dispute. We shall assume, but purely for argument's sake, that the entire [Oracle] Java API falls within the definition of that which can be copyrighted. We shall ask instead whether Google's use of part of that API was 'fair use.'[31]

Analysing the question of whether Google's reproduction of certain interfaces[32] from Oracle's Java SE program was fair use, the US Supreme Court ultimately decided that Google's reproduction fell within the 'fair use' provisions of US copyright law.[33] Although the decision is complex, and goes through a detailed analysis of all the statutory fair use factors, perhaps the two most important aspects of the decision analysing whether the use of software interfaces are subject to copyright claims by the authors of those interfaces related to the questions of 'transformative use' and the manner in which future courts may decide fair use questions in software copyright disputes. On the question of 'transformative use', the Court in *Google v Oracle* stated:

> Here, Google's use of the [Oracle] Java API seeks to create new products. . . . To the extent that Google used parts of the [Oracle] Java API to create a new platform that could be readily used by programmers, its use was consistent with the creative 'progress' that is the basic constitutional objective of copyright itself.[34]

Thus, it could well be that, under the *Google v Oracle* fair use test, most reproductions of software interfaces in the US in order to create new, interoperable but non-competitive software will be found to be not subject to the copyright claims of the authors of those interfaces, because such uses will be determined to be 'transformative'. If so, the outcome in the US on software interfaces may not be much different than the outcome in the EU under the *SAS Institute* decision. Future decisions in

[31] *Google LLC v Oracle America, Inc.*, 593 US ___, 141 S. Ct. 1183, Docket No. 18-956, Opinion of the Court at 15 (5 April 2021). The dissenting opinion in that decision criticised the majority's failure to address the copyrightability question. *Google LLC v Oracle America, Inc.*, see note 34, Dissenting Opinion at 1–2.

[32] Although the *Google v Oracle* decision often refers to 'Java APIs' and 'interfaces', the discussion focuses exclusively on Google's reproduction of the method, class, and package structures used in certain parts of Java most useful in smartphones for programers familiar with the schema in Java. *Google LLC v Oracle America, Inc.*, 593 US ___, 141 S. Ct. 1183, Docket No. 18-956, Opinion of the Court at 26 (5 April 2021).

[33] *Google LLC v Oracle America, Inc.*, 593 US ___, 141 S. Ct. 1183, Docket No. 18-956, Opinion of the Court at 1 (5 April 2021).

[34] *Google LLC v Oracle America, Inc.*, see note 37, at 25.

the US applying the fair use analysis to different software interface scenarios may clarify if this is indeed the case.[35]

The *Google v Oracle* decision also clarified an important point on 'fair use' analyses in copyright disputes:

> 'In this case, the ultimate "fair use" question primarily involves legal work. "Fair use" was originally a concept fashioned by judges … Our cases still provide legal interpretation of the fair use provision. And those interpretations provide general guidance for future cases.'[36]

This finding is an important procedural point in US law as it may result in early hearings in software copyright disputes to resolve the 'primarily legal' question of fair use. Early hearings on the legal question of patent claim interpretation have now become a regular part of patent litigation practice in the US.[37] Such procedures, if adopted in the future, could have curtailed the nearly eleven-year litigation history of the *Google v Oracle* dispute. Nevertheless, unless there are future developments in the US on the question of software copyrights in general, or software copyright interfaces in particular, there will be greater ambiguity about the use of such interfaces, and the effort required to resolve them, in the US than there currently is in the EU.

The question of what parts of software are subject to copyright protections and which are free for anyone to use without fear of a claim of copyright infringement or licence violation is an important issue when analysing the effect of Open Source licences on downstream recipients. This is particularly the case with copyleft licences, as the downstream recipient may not need to follow the licence's requirement to use the same licence, if they are using only material for which either copyright protection does not extend, or for which the right of 'fair use' or 'fair dealing' (or equivalents) applies. In the EU, the *SAS Institute* case provides some concrete guidance on this question; the courts in the US have not resolved this issue as conclusively, as the US Supreme Court's decision in *Google v Oracle* left the question to a case-by-case, legal analysis somewhat dependent upon underlying facts. As a result, distributors

[35] An early, and thoughtful, analysis of this question—written after the *SAS Institute* decision but almost a decade before the ultimate decision in *Google v Oracle*—may be found at Walter van Holst, 'Less May Be More: Copyleft, -Right and the Case Law on APIs on Both Sides of the Atlantic' (2005) 5(1) *Journal of Open Law, Technology & Society* at 5 <https://jolts.world/index.php/jolts/article/view/72/143> accessed 6 April 2021. In that analysis, the author concludes that ' [w]hen taking the most recent jurisprudence on software APIs [i.e., *SAS Institute*] into account, one can argue that the LGPL is not really the Lesser GPL, but that the GPL is based on a by now outdated understanding of software copyright and effectively becomes equal to the LGPL'. Von Holst, 'Less May Be More', at 13. This conclusion may also follow if the *Google v Oracle* decision is ultimately determined to be very near to the outcome in *SAS Institute*.

[36] *Google LLC v Oracle America, Inc.*, 593 US ___, 141 S. Ct. 1183, Docket No. 18-956, Opinion of the Court at 19 (5 April 2021).

[37] Edward Brunet, 'Markman Hearings, Summary Judgment, and Judicial Discretion' (2005) 9(1) *Lewis & Clark Law Review* 93, at 95–96.

and recipients of code under Open Source licences may be forced to analyse the question of whether the use of software interfaces require adherence to the Open Source licence terms depending on whether the use is within, or without, the US. If the use is in the US, a detailed factual analysis of the circumstances in which software interfaces are reproduced, under the fair use tests under US copyright law,[38] may be required—at least until such time as future court decisions provide greater guidance on when reproduction of software interfaces is not 'fair use'.[39]

3.1.3.2 Functional dictation and merger doctrine

Authored material that would otherwise be copyrightable expression has nevertheless been found not to be subject to copyright protection when the expression is directed to an idea that is incapable of being expressed, as a practical matter, in more than one or a small number of ways. In the UK, this concept has been established under the rationale 'that if expression is dictated by technical function then the criterion of originality [required for copyright protection] is not satisfied'.[40] In the US, this concept is designated the 'merger doctrine'—that the copyrightable expression in a work of authorship has 'merged' with the non-copyrightable idea being expressed, when that idea is incapable of being expressed, as a practical matter, in more than one or a small number of ways.[41] Although in many ways non-copyrightability and functional dictation/merger doctrine cover similar territory and rely on related statutory bases and case law, the latter doctrine does admit of the possibility of arguing that—even though there may be multiple ways of expressing certain ideas, facts, systems, or processes through code—to the extent that those multiple ways are unduly constraining on the ability of other authors to capture those ideas, facts, systems, or processes without running afoul of a different author's copyrights, protection under copyright should not be afforded.

[38] 17 USC § 107.

[39] Although the newness of the *Google v Oracle* decision has meant there is limited commentary on its potential import on software development practices in the US, at least one commenter has indicated that reproduction of code for interoperability may be found to be fair use given recent court decisions, including *Google v Oracle*, emphasising transformative uses of copyrighted materials as being more often than not fair. 'Google's Supreme Court win could actually benefit the little guy' MarketPlace (6 April 2021) <https://www.marketplace.org/shows/marketplace-tech/googles-supreme-court-win-could-actually-benefit-the-little-guy-oracle-java/> accessed 6 April 2021.

[40] See *SAS Inst. Inc. v World Programming Ltd.* [2013] EWCA Civ 1482, [31]–[33], available at <http://www.bailii.org/ew/cases/EWCA/Civ/2013/1482.html> accessed 13 April 2022.

[41] See Pamela Samuelson, 'Reconceptualizing Copyright's Merger Doctrine' (2017) 63 *Journal of the Copyright Office Society of the United States of America* 417–70, available at <https://papers.ssrn.com/sol3/Delivery.cfm/SSRN_ID2763903_code1160955.pdf?abstractid=2763903&mirid=1&type=2>. Professor Samuelson argues that merger doctrine in the US is not just limited to preventing 'ideas' from being captured and restricted as copyrightable expression, but that instead there are other dichotomies that the merger doctrine also covers, including 'fact/expression', 'process/expression', 'system/expression', Samuelson, 'Reconceptualizing Copyright's Merger Doctrine' at 417–70. Each of these dichotomies represent a possible limitation on the ability to claim copyright protection on certain aspects of computer software.

3.1.3.3 'Fair dealing' and 'fair use'

The law in both the UK and the US allows for certain 'fair' exercises of the exclusive rights of a copyright holder without subjecting those exercises to infringement liability. In the UK, this exception to the holder's rights is called 'fair dealing';[42] in the US, the exception is called 'fair use'.[43] Despite the similarities in the names, the effect—particularly in the area of software—can be quite different. In the US, 'fair use' has been the foundation of a number of defences to claims of copyright infringement for software,[44] and in fact fair use was the basis upon which the *Oracle v Google* dispute was decided by the Supreme Court of the US.[45] Fair use as a defence to a claim of copyright infringement in the US is defined in statute and is analysed using a multifactored factual analysis not specific to the use that is being made; a court must analyse:

- the purpose and character of the use, including whether such use is of a commercial nature or is for non-profit educational purposes;
- the nature of the copyrighted work;
- the amount and substantiality of the portion used in relation to the copyrighted work as a whole; and
- the effect of the use upon the potential market for or value of the copyrighted work.[46]

In contrast, fair dealing in the UK is much more specific to the type of use being made, and specifically excludes many uses of computer software from its ambit.[47] Instead of relying upon fair dealing to address certain 'fair' uses of computer software, the UK Copyright Act sets forth, in sections separate from the defined uses under fair dealing, specific acts relative to computer software which are explicitly excluded from infringement:[48]

- making any back up copy;
- converting it into a higher-level language;
- decompiling it to obtain the information necessary to create an independent program which can be operated with the program so decompiled, or with another program;

[42] UK Copyright Act, see note 18, §§ 29.
[43] 17 USC § 107 (1992).
[44] See, e.g., *Sega Enterprises Ltd v Accolade Inc.*, 977 F.2d 1510 (9th Cir. 1992).
[45] *Google LLC v Oracle Am., Inc.*, 593 US ___, Docket No. 18-956, Opinion of the Court at 1 (5 April 2021).
[46] 17 USC § 107 (1992).
[47] UK Copyright Act, see note 18, § 29 (excluding converting a computer program to a higher-level language, copying incidental to such a conversion, and observing, studying or testing the functioning of a computer program in order to determine the ideas and principles which underlie any element of the program).
[48] UK Copyright Act, see note 18, §§ 50A–50C.

- observing, studying, or testing the functioning of the program in order to determine the ideas and principles which underlie any element of the program;
- copying it or adapting it for the purpose of correcting errors in it.

In summary, US law admits of perhaps a broader range of acts concerning software that might be considered 'fair' and outside of the threat of infringement claims, but in order to establish that such use is 'fair', the accused infringer is subject to a detailed factual analysis amenable to differing interpretations. In the UK, in contrast, the activities considered 'fair' are much more narrowly circumscribed, but are potentially much easier to establish factually.

3.1.4 Derivative works in software copyright

3.1.4.1 A brief summary of computer software architecture and interactions

The many ways in which a particular piece of software can be configured to interact with other pieces of software is far beyond the scope of this chapter, and is sufficiently malleable with new developments in technology, that it would be virtually impossible to discuss all the different ways such interactions can occur, and the licensing and copyright implications that result. Nevertheless, the process of writing source code, converting that source code into executable code, delivering that executable code to a target computing device, and executing that executable code on that target device, often fall into certain general techniques that it is of use to describe them in brief to discuss common use cases that frequently raise questions with regard to the effect of certain Open Source licences. Figure 3.1 represents a common software build scenario.

One or more modules A and B are written in source code by human programers. A software tool called a compiler is used to both convert the source modules into a format called object code, and (possibly) to intermingle parts of A and B together into a unitary object module AB. There may also be standard libraries or other pre-existing code modules which the object module AB is designed to make use of.

These libraries can be used by the compiled code AB in one of two ways—via a static link or a dynamic link. In a static link, the object code AB is input into a tool called a linker, together with object code of the library SL, to form a unitary binary ABSL, all of which is distributed to a user to be executed or run on their computing machine. In a dynamic link, the object code AB is distributed to the user under the assumption or expectation that the library which the object code AB is designed to use is already installed on the user's computing machine or will be separately obtained by the user. The combination of AB and the library DL is not made until 'run-time', in other words the two are combined together in the computing machine's memory upon execution of the object file AB.

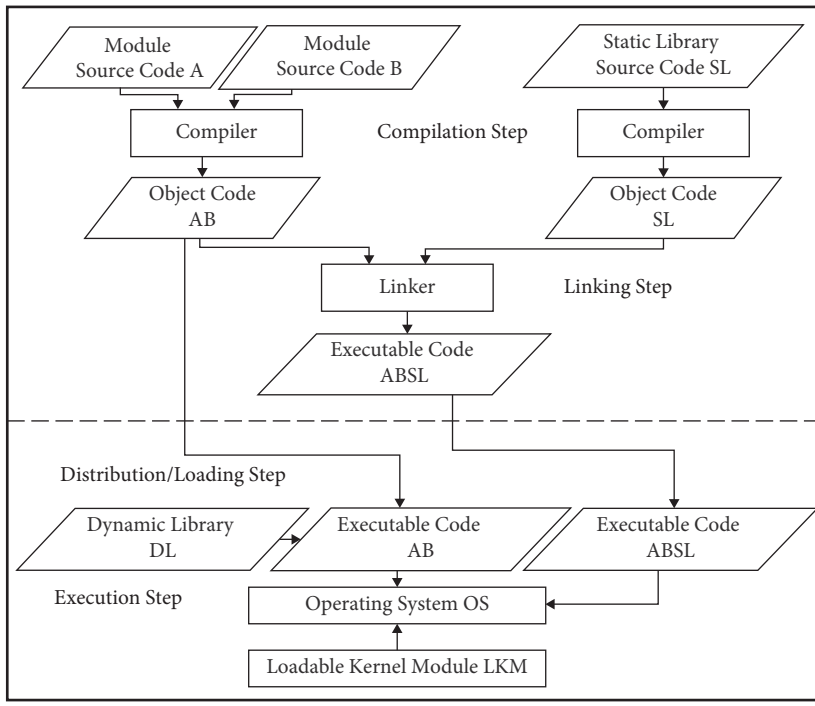


Figure 3.1 Common Software Build Scenarios

There is an additional piece of software that may be used for which free and open source licensing issues are implicated. A loadable kernel module (often abbreviated as LKM) is a piece of code (in most cases, drivers written to allow the kernel of the operating system—which manages the computing machine's resources—to make use of certain components within the computing machine) that is dynamically loaded into the operating system kernel at execution, rather than integrated into the kernel itself.

The concepts outlined above may be of importance to understand when interpreting the impact of certain free and open source licensing, particularly the copyleft licences which purport to apply their terms to programs which may be combined or linked, as discussed in more detail in section 3.2.2 later in this chapter.

3.1.4.2 Derivative works and Open Source

A number of Open Source licences use the term 'derivate works' to define the scope of certain activities related to the modification, adaptation, or translation of the licensed code.[49] As discussed earlier in section 3.1.2, 'derivative works' is

[49] See, *inter alia*, GPLv2, § 0.

a concept from US copyright law which encompasses things like modification, translation, adaptation, and arranging in international copyright treaties or non-US national laws, but it does not have universal meaning that is consistent across national laws,[50] or even within courts in the US.[51] The boundaries of what is or is not a derivative work, or is equivalent or corelated in non-US law, is also circumscribed by the exceptions to copyright protection and enforceability set forth in section 3.1.3 earlier in this chapter. Because of this, any interpretation of the rights granted under an Open Source licence, or the obligations attached to that grant, will be subject to potential ambiguity to the extent it references derivative works without further clarifying what types of derivative works are governed by the terms of the licence. More detail on how that ambiguity works in practice can be found in section 3.2.2 later in this chapter.

3.2 Forms of Open Source Licensing

As discussed in more detail in Chapter 1, Open Source licensing is—at its heart—a philosophical movement which seeks to upend or reverse the more conventional 'proprietary' model where the rights required to make productive use of some useful thing are granted restrictively, and in most cases without any rights to study, learn, and adapt the underlying architecture or design of that thing. In order to put into practice this philosophy, there must be legal rights granted, and those rights need to be granted in a way that is consistent with that philosophy. One of the most important forms of legal rights used to achieve this aim is copyright—specifically, the copyright in both the source and executable forms of software discussed in more detail in section 3.1.2 earlier in this chapter.

In the late 1980s,[52] two alternative ways of achieving the overall aims of the free and open source philosophy were (roughly) simultaneously created: permissive licensing and copyleft licensing. Each of these models have numerous variants, and at least copyleft has two generally recognised sub-variants—'strong' copyleft and 'weak' copyleft, but both models share certain common characteristics and in certain cases can be used in a complementary way. Practitioners are nevertheless cautioned that there are also significant incompatibility issues between some of these

[50] Till Jaeger, 'Enforcement of the GNU GPL in Germany and Europe' (2010) 1 *Journal of Intellectual Property, Information Technology and E-Commerce Law* 34, at 36.

[51] Omar Johnny, Marc Miller, and Mark Webbink, 'Copyright in Open Source Software—Understanding the Boundaries' (2010) 2(1) *Journal of Open Law, Technology & Society* 13, at 24.

[52] The earliest examples of licences satisfying both the Free Software Definition and the Open Source Definition, the MIT licence, the BSD licence, and the GPL licence, date—at their earliest instantiation—from 1987, 1988, and 1989, respectively. See Gordon Haff, 'The Mysterious History of the MIT License' opensource.org (26 April 2019), <https://opensource.com/article/19/4/history-mit-license> accessed 19 January 2020; Richard Stallman, 'New General Public License' (February 1989) <https://groups.google.com/forum/#!msg/gnu.announce/m0Jjj_64PeQ/8xL1xkVKJb8J> accessed 9 March 2020.

licences, and understanding when and where particular licences can be used in a complementary way, and where such licences create unresolvable conflicts, is very much dependent upon the text of the particular licences used, the state of copyright law in the particular jurisdiction where the licence might be enforced, a good understanding of the particular programing paradigm being used, and how that programing paradigm maps to the licence texts at issue and copyright law in the particular jurisdiction in question.

Most free and open source licences allow the user to make 'private' use of the software—meaning that anything the user does without allowing access to others imposes no legal obligations on that user.[53] It is when the user seeks to allow others access to that software—typically by distributing the software, or a modified version of the software—that the licence imposes legal obligations on that user. Many of those obligations are similar amongst all licences, whilst others differ in standard ways that allow categorisation and subcategorisation between licences. Those categorisations and sub-categorisations are discussed in more depth in the following sections.

3.2.1 Permissive licensing

3.2.1.1 The BSD and MIT licences

The BSD[54] and MIT[55] licences are the oldest free and open source licences still in use to this day. Both are quite similar in the way that they are structured and the obligations that they impose upon the recipient, although there are some subtle differences that might cause an author to choose one over the other, or for a court or arbiter to determine that they have different legal effect. Both licences grant broad licences, at least under copyright. Both oblige preservation of copyright notices for those who exercise the licence grants. Both oblige that a copy of the licence text be provided for any exercise of the licence grants. Both disclaim liability on behalf of the authors.

[53] The Free Software Foundation's so-called Freedom Zero— 'The freedom to run the program as you wish, for any purpose'—generally contemplates this concept. See Free Software Foundation, 'What is Free Software? The Free Software Definition' <https://www.gnu.org/philosophy/free-sw.html.en> accessed 29 February 2020. As noted later in the chapter, certain Open Source licences test this concept.

[54] There is no single 'BSD' licence; there are numerous variants, generally differentiated by the number of clauses they contain. Thus, 4-Clause BSD <https://spdx.org/licenses/BSD-4-Clause.html>, 3-Clause BSD <https://spdx.org/licenses/BSD-3-Clause.html>, 2-Clause BSD <https://spdx.org/licenses/BSD-2-Clause.html>, 1-Clause BSD <https://spdx.org/licenses/BSD-1-Clause.html>, and 0-Clause BSD <https://spdx.org/licenses/0BSD.html> accessed 21 July 2022. The 4-Clause BSD contains an 'advertising clause' which is generally considered to create compliance issues, and is thus generally disfavoured. See Sinclair, 'Licence Profile: BSD', see note 21, at 4–5.

[55] There is also some dispute as to whether there is more than one 'MIT' licence. See GNU Operating System, 'Various licenses and comments about them', <https://www.gnu.org/licenses/license-list.html#Expat> accessed 10 March 2020. The version approved by the Open Source Initiative is generally considered the canonical version, and is the version discussed later in the chapter. See <https://opensource.org/licenses/MIT>.

The effect of both licences is to allow recipients to exercise the author's copyright rights, without imposing any obligations on the recipients to use the same licence for any further downstream distribution.[56] They are thus *permissive* in the way in which they allow alternative licensing models to be exercised by recipients—including, restrictive, 'proprietary' licensing, and copyleft licensing.

One distinction between the two licences is that the MIT license expresses its grant using a non-statutory term ('deal in'), and then recites US statutory verbs as non-limiting examples: '[p]ermission is … granted … to *deal in* the Software without restriction, *including without limitation* the rights to use, copy, modify, merge, publish, distribute, … and/or sell copies' (emphasis added), whereas the BSD license uses fewer verbs, and uses them directly in the grant: '*[r]edistribution* and *use* …, *with or without modification*, are permitted' (emphasis added). Another distinction is that the MIT license explicitly allows sublicensing, whereas the BSD license, if it does so, does so indirectly—using the term 'redistribution'. The extent to which these distinctions are meaningful has yet to be adjudicated, and the two licences are generally considered to be relatively interchangeable in the permissions they give and the obligations they impose on those permissions.

3.2.1.2 The Apache licence

In 2000, the ASF published an alternative form of a permissive licence; that licence is now on its second—and by far most commonly used—iteration, the Apache Software License 2.0 (the Apache 2.0). The Apache 2.0 license, drafted more than a decade after the BSD and MIT licenses came into being, was designed to address some perceived ambiguities or missing features in those two licences, in particular a more robust set of definitions of the licence grants themselves[57] (and the parties to whom they extend) express granting of patent rights (with the inclusion of a patent 'defensive suspension' clause to revoke patent rights to those entities taking assertive patent actions against the licensed software) and other features designed to make the licence terms more consistent with generally

[56] There is a fairly small minority of users of these licences who have argued that the requirement to *provide* a copy of the licence obliges the recipient to *use* that licence, thus rendering these licences copyleft. See Anonymous Coward, 'Theo de Raadt on Relicensing BSD Code', *OpenBSD Journal* (13 September 2007) <http://undeadly.org/cgi?action=article&sid=20070913014315> accessed 13 April 2022. The general consensus is, however, to the contrary and re-licensing under different terms—including restrictive proprietary licences and copyleft licences—is a common practice with permissively licensed software.

[57] Of particular interest is the manner in which the Apache 2.0 licence defines the scope of modifications/derivative works that are subject to the Apache 2.0 licence terms. The Definition section of the Apache 2.0 licence states that '[f]or the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.' Given the overall permissive nature of the licence, this clarification of the scope of the grant may be in practice relatively insubstantial, although it could be an important distinction in some architectural scenarios given that avoiding the application of the Apache 2.0 licence to certain works may be useful because of the incompatibility of the Apache 2.0 licence with certain of the GNU family of licences. See section 3.3.2.3 later in this chapter.

accepted licence drafting standards. Nevertheless, the overall effect of the copyright licence grants in the Apache 2.0 licence are intended to be the same as those of the BSD and MIT licenses discussed earlier, requiring preservation of copyright notices and providing a copy of the licence with the code, but otherwise permissively allowing the recipient of the code to use terms of its own choosing, including proprietary licensing, or copyleft licensing, when that recipient exercises the granted copyright licences. As discussed in more detail in section 3.3.2.3, there are certain caveats to the overall permissive nature of the Apache 2.0 licence: the GNU family of licences. The Free Software Foundation (FSF) has stated that the Apache 2.0 license is incompatible with GPLv2;[58] although the ASF disagrees with this assessment, it nevertheless states that with regard to use of GPLv2, 'you should always try to obey the constraints expressed by the copyright holder when redistributing their work'.[59] This incompatibility introduces certain complications in architecting software stacks that may include Apache 2.0 and GPLv2 code (in particular, the Linux kernel of the GNU/Linux operating system, which is licensed under GPLv2), and could very well be the reason why the more recent, and more carefully drafted, Apache 2.0 license has not supplanted the continued popularity of the BSD and MIT licenses as *de facto* choice when selecting a permissive licence.

3.2.1.3 Other permissive licences

There are a large number of permissive licence variants currently listed as either approved by the Open Source Initiative (OSI) or determined to be free software licences by the FSF. The vast majority of these licences are variants of the BSD (which itself has numerous variants) or MIT licenses, and a handful of Apache 2.0 variants as well, and thus would generally operate in a manner similar to those licences. Nevertheless, when confronted by these numerous variants, it is advisable to read their terms carefully as some variants may introduce additional complications or incompatibilities.[60] Two other permissive licences that have some degree of popularity and which practitioners may encounter and thus need to familiarise themselves with are: the Academic Free License[61] and the Artistic License.[62] The

[58] See FSF Licence Comments, see note 59, <https://www.gnu.org/licenses/license-list.html#apache2> accessed 21 July 2022. Note that by the logic used in the FSF's commentary on the Apache 2.0 licence, Apache 2.0 is likely also to be incompatible with the Affero GPLv1.0 and 2.0 licences, and partially incompatible with the Lesser GPLv2.1 licence.

[59] Apache Software Foundation, 'GPL Compatibility' <https://www.apache.org/licenslicences/GPL-compatibility.html> accessed 10 March 2020.

[60] For example, licences with so-called advertising clauses may be incompatible with many other Open Source licences. See GNU Operating System, 'The BSD License Problem' <https://www.gnu.org/licenses/bsd.html> accessed 10 March 2020.

[61] Open Source Initiative, 'Academic Free Licence ("AFL") v. 3.0' <https://opensource.org/licenses/AFL-3.0> accessed 10 March 2020.

[62] Open Source Initiative, 'Artistic Licence v. 2.0' <https://opensource.org/licenses/Artistic-2.0> accessed 10 March 2020.

Academic Free License is designed to be somewhat similar in operation to the Apache 2.0 license in that it has more robust drafting and has specific reference to patent rights; it is also notable in that its author has explicitly stated that that licence is designed to operate as a contract,[63] thus explicitly settling the 'bare licence versus contract' debate—discussed in section 3.4 later in this chapter—for that licence. The Artistic Licence is notable in that it was the subject of one of the first, and still one of the leading, cases in the US examining the operation of Open Source licensing.[64] That decision, at a minimum, demonstrates that although permissive licences are designed to be easy to comply with and to allow flexibility to the recipient in modifying and combining code while also being able to choose its own licence, failure to comply with even the simplest licence conditions in a permissive licence can result in enforcement action and could be found to be a violation of the author's copyright rights.[65]

3.2.2 Copyleft licensing

The second broad category of Open Source software license types are the copyleft[66] licences. In all but one very important way, copyleft licences are designed to operate in a way similar to the permissive licences—a broad grant of copyright rights (and in virtually every copyleft licence, at least an attempt to do the same with patent rights), a requirement to preserve copyright notices, and a requirement to provide a copy of the licence with the code. The important distinction is that copyleft licences require certain exercises of the author's copyright rights to be licensed under identical terms. Thus, copyleft licences put limitations on the recipient's ability to use other licensing models (e.g. proprietary, or permissive, or even a different copyleft licence) for their own downstream activities. Copyleft licences are often broken into two subcategories: 'strong' copyleft and 'weak' copyleft. 'Strong' copyleft licences, in general, are designed to have fewer exclusions[67] to the requirement to use the same licence when exercising the author's copyright rights, whereas

[63] Lawrence Rosen, 'Open Source Licensing: Software Freedom and Intellectual Property Law' (Prentice Hall: Upper Saddle River, NJ, 2005) at 181.

[64] *Jacobsen v Katzer*, 535 F.3d 1373 (Fed. Cir. 2008).

[65] *Jacobsen v Katzer*, note 64, at 1382.

[66] 'Copyleft' is a coined term intended to be a pun on 'copyright', contrasting the highly permissive terms of that family of licences with the generally perceived highly restrictive nature of copyright protection and copyright licensing in the software industry. An additional pun for copyleft, 'all rights reversed', is a play on the now-obsolete notification that used to often accompany copyrighted works, 'all rights reserved'.

[67] Although strong copyleft licences are generally designed to require the use of the same licence when the recipient exercises the author's copyright rights, even the strongest of copyleft licences (GPLv2, GPLv3, AGPLv1, and AGPLv3) all allow the recipient to engage in some form of private modification of the code without triggering the obligations of those licences. See, e.g., GPLv3 § 0 (definition of 'propagate').

'weak' copyleft licences are designed specifically to articulate circumstances where an exercise of the author's copyright rights is permitted while allowing for alternative licensing (again, be it permissive, or proprietary, or a different copyleft licence) of code resulting from that exercise. The strong copyleft licences typically require that derivative works (or some other formulation intended to capture that concept under both the law of the US and other jurisdictions) cannot be distributed under any other licence.

3.2.2.1 The GNU family of licences

Copyleft licensing was pioneered by—and the term 'copyleft' was indeed coined by—the FSF. The FSF maintains a family of copyleft licences—all under the 'GNU'[68] moniker—all intended to allow different forms of copyleft to be used, depending on the degree of copyleft obligation to be imposed on the user. This family of licences can generally be divided into the 'strong copyleft'—GPLv2, GPLv3, AGPLv1 and AGPLv3—and 'weak copyleft'—LGPLv2.1 and LGPLv3.0—variants.

3.2.2.1.1 The GNU General Public Licence (GPL)

3.2.2.1.1.1 GPLv2 The GNU General Public License, version 2 ('GPLv2') is a strong copyleft licence first published by the FSF in 1991. It grants the right to exercise enumerated copyright rights (copy, distribute, and modify the software), under the condition that the resulting software is again distributed[69] under the identical conditions of GPLv2. As with other Open Source licences, GPLv2 requires including the GPLv2 licence text, providing the source code for any distributed executables, and making reference to the disclaimer of warranty. GPLv2 states that failure to follow the licence terms results in the revocation of the licence, although third parties (such as downstream recipients) are unaffected by such failure. The obligation to grant access to the source code in case of distribution of executable copies requires providing 'Complete Corresponding Source Code', defined as 'all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable'.[70]

GPLv2, by its own terms, extends the obligations to a 'work based on the Program', defined as 'any derivative work under copyright law; that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language'.[71] However, GPLv2 also states that 'mere

[68] 'GNU' is a recursive acronym for 'GNU's not UNIX', an express acknowledgement that the FSF was attempting to create—using the GNU family of licences—a computer operating system intended to be an alternative to the then-ubiquitous UNIX operating system.

[69] It is notable that the licence obligations in GPLv2 are attached upon 'distribution' of code covered by that licence; the licence itself makes clear that merely running the program can be done without need to follow the licence obligations. See GPLv2, § 0.

[70] GPLv2, § 3. This detailed definition is intended to prevent source distributions which are not amenable to modification and compilation because of scripts or tools not otherwise available to distributees.

[71] GPLv2, § 0.

aggregation of another work not based on the Program with the Program … on a volume of a storage or distribution medium does not bring the other work under the scope of this License'.[72] GPLv2 further states that when sections of the new work

> are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.[73]

GPLv2 states that the licence is designed 'to exercise the right to control the distribution of derivative *or collective works* based on the Program';[74] thus any analysis of whether GPLv2 would apply to a particular use and distribution of code under that licence would need to consider the question of whether a derivative work or collective work has been created.[75]

As may be clear from the earlier discussion, determining what particular exercises of copyright rights under GPLv2 obligate the licensor to follow the terms of that licence is not explicitly clear; this issue is made even more thorny by the fact that which does, and does not, fall outside of the realm of 'derivative works' under US Copyright law (from whence the term 'derivative works' in the licence comes) for software is not well-defined and in fact subject to a number of different tests.[76]

GPLv2's obligations purport to apply to any derivative or collective works; anything that is outside the definition of a derivative or collective work should not be affected by the licence's obligations. According to the FSF—stewards of GPLv2—a work that links to code licensed under GPLv2 (statically or dynamically) forms part of the modified work and it must be treated accordingly.[77] This assertion is a matter of some controversy, and many commentators believe that although static linking arguably does require the linked program to be licensed under GPLv2, dynamic linking should not.[78]

[72] GPLv2, § 2.
[73] GPLv2, § 2.
[74] GPLv2, § 2.
[75] 'Collective works' are a concept under US Copyright Law, 17 USC § 101, defined as 'a work … in which a number of contributions, constituting separate and independent works in themselves, are assembled into a collective whole'. GPLv2 does not make clear the distinction between covered 'collective works' and uncovered 'mere aggregations'.
[76] See section 3.1.4.2 of this chapter.
[77] See Free Software Foundation, 'Frequently Asked Questions About the GNU Licenses' <https://www.gnu.org/licenses/gpl-faq.en.html#GPLStaticVsDynamic> accessed 29 February 2020.
[78] See, e.g., Malcolm Bain, 'Software Interactions and the GNU General Public License' (2010) 2(2) *Journal of Open Law, Technology & Society* 165, at 177.

In the end, the GPLv2 licence in part depends on the degree of current ambiguity around the scope of derivative works (or, in other jurisdictions, the acts of adapting/arranging/translating of copyrighted works). As more decisions are rendered interpreting the scope of copyright in software—such as, for example, whether certain interfaces in software are copyrightable at all[79]—or the specific provisions of 'work based on the Program' under GPLv2, more clarity may be provided as to how broadly that licence applies to other programs that may work with or around GPLv2 licensed code.

3.2.2.1.1.2 GPLv3 In 2007, the FSF published the GNU General Public License, version 3.0 (GPLv3).[80] GPLv3 was intended to modernise GPLv2 in several ways considered to be important to the FSF, including: (i) 'internationalising' the language to make it less US law-centric; (ii) adding clearer and more specific conditions and obligations around patent rights; and (iii) addressing certain behaviours considered to be antithetical to software freedom, but which it was believed were not addressed adequately in GPLv2.[81]

Many of the same rights and obligations that existed in GPLv2 were preserved in GPLv3, albeit with slightly updated language; thus, much of the discussion of GPLv2 in section 3.2.2.1.1 earlier in this chapter would apply equally to GPLv3.[82] GPLv3 does explicitly state that it intends to capture, under the obligation to provide source code, 'dynamically linked subprograms that the work is specifically designed to require',[83] thus making explicit in the licence text that which was only referenced in a FSF FAQ for GPLv2—that dynamic linking was considered by the FSF to require the program so linked to be licensed under GPLv3. As discussed earlier in section 3.2.2.1.1 with reference to GPLv2, whether dynamic linking would—under copyright law—require the linked program to be licensed according to the terms of the program to which it is linked is a matter of some dispute that is currently unresolved.

GPLv3 also attempts to address concerns raised by the passage of the Digital Millennium Copyright Act (DMCA)[84] in the US in 1998—an Act not in effect

[79] See Section 3.1.3.1 above.

[80] GNU Operating System, 'GNU General Public License' <https://www.gnu.org/licenses/gpl-3.0.html> accessed 29 June 2007.

[81] See Free Software Foundation, 'GPLv3 First Discussion Draft Rationale' (16 January 2006), <http://gplv3.fsf.org/gpl-rationale-2006-01-16.pdf> accessed 13 April 2022.

[82] One change between GPLv2 and GPLv3 that may be merely a wording clarification to capture international norms is that the licence obligations in GPLv3 are triggered upon 'conveyance', a defined term which includes an embedded defined term, 'propagation', which term includes, but is not limited to, distribution. See GPLv3, § 0. The definition of 'propagation' in GPLv3 references both direct and secondary liability under copyright law, and thus attempts to capture acts that would only be infringing on licensor's rights upon acts by third parties. The legal effect of this definitional change is as-yet undetermined, although it likely is an attempt—together with the definition of 'Corresponding Source' in § 1—to cover dynamic linking.

[83] GPLv3, § 1.

[84] 17 USC § 1201 (1999).

at the time of publication of GPLv2 in 1991. GPLv3 provides that no work covered by the licence may be deemed part of an effective technological protection measure under any applicable law;[85] since 'technological protection measures' is a term from the DMCA, this provision is aimed squarely at that law and any related laws outside the US, attempting to allow copyright holders a right of action against those circumventing various technical means designed to prevent access to the copyrighted works, such as Digital Rights Management (DRM). Given all the other obligations of GPLv3 to provide Complete Corresponding Source, as well as to license under the terms of GPLv3, it is difficult to foresee scenarios where a technological protection measure could be built into GPLv3 code to adequately invoke the DMCA or related laws. Nevertheless, the drafters of GPLv3 were sufficiently concerned that users might attempt to do so that they made explicit that it was prohibited by that licence.

GPLv3 also includes a requirement intended to prevent certain hardware manufacturers from using GPLv3 code on their devices but including technical mechanisms in that hardware—such as installation keys—to prevent the hardware user from modifying, installing, and running the GPLv3 code on that device. The 'Installation Information'[86] section of GPLv3 is complex and admits of several exceptions whereby a hardware maker would not be required to comply. Perhaps the most important exception is that it only applies to a certain subsegment of hardware devices, so-called User Products. The definition of a 'User Product' for which 'Installation Information' must be provided under GPLv3 is derived from a definition in consumer protection laws in the US;[87] the interpretive law surrounding that consumer protection law—and possibly corresponding consumer protection laws outside the US—is likely to guide a court in understanding the sorts of devices to which that provision applies. For such products, encryption keys, hardware checksums, or other technical information needed to install and operate modified GPLv3 software on such products would need to be provided as 'Complete Corresponding Source'.[88]

3.2.2.1.2 The GNU Lesser General Public Licence (LGPL)

The FSF realised that in certain circumstances, it would not be pragmatic to license all code under a 'strong' copyleft licence like one of the GPL licences. For example, certain libraries might be of greater use, and enjoy greater adoption, if they did not impose copyleft requirements on any program making use of that library. As a result,

[85] GPLv3, § 3.
[86] GPLv3, § 6.
[87] 15 US Code § 2301(1) (1975).
[88] For a detailed explanation of how the 'Installation Information' requirement of GPLv3 works and how it relates—if at all—to requirements in GPLv2, see P McCoy Smith, ' "Installation Information", GPLv2 and GPLv3: What is it and what must you provide?' (27 September 2021) Linux Foundation Open Source Summit <https://www.youtube.com/watch?v=6W3LBlkOpDM&t=2s> accessed 8 June 2022.

the FSF created a licence—initially named the 'Library General Public License' but later changed to be called the 'LGPL'—designed to be a 'weak' copyleft version of GPL. Like GPL, there exist two common versions in current use—LGPLv2.1, and LGPLv3. The text of LGPLv2.1 corresponds, with notable exceptions, to the text of GPLv2, whereas the text of LGPLv3 corresponds, again with notable exceptions, to GPLv3. As such, much of the discussion above with regard to GPLv2 and GPLv3 would apply to their counterpart LGPL versions.

The first exception—found in both LGPLv2.1 and LGPLv3—states that an application may make use of the following header file elements of code licensed under LGPL without the requirements of LGPL applying to that application: 'numerical parameters, data structure layouts and accessors, or small macros, inline functions and templates (ten or fewer lines in length)'.[89] This first exception recognises that there may be a need for some limited use to be made of parts of those header files in order for an application to interoperate (or to be statically linked) with the LGPL code, but that such uses should not force the application itself to be subject to the obligations of LGPL. This exception thus provides a form of a defined *de minimus* or fair use/fair dealing exception to the copyleft obligations of LGPL.

The second exception—also found in both LGPLv2.1 and LGPLv3—allows the creation of 'Combined Works' (a work 'produced by combining or linking' an application with the LGPL code, which Combined Work may be licensed under 'terms of your choice', i.e., any licence terms, not just LGPL).[90] The only limitation on this exception is that the 'terms of your choice' do not 'effectively … restrict modification of the portions of the Library contained in the Combined Work and reverse engineering for debugging such modifications', together with requiring the source code for the LGPL portion of the Combined Work, and a notification that LGPL applies to that portion. This second exception recognises that many libraries, or other commonly-referenced software components, will need to be combined together—such as by linking—and to oblige the entire combined work to be licensed under LGPL would decrease the use of such libraries. Thus, the second exception is designed to allow any licence choice (copyleft, permissive, or even proprietary), as long as the recipient of the combined work has the ability to modify and recombine the LGPL code, and to do effective debugging of the modified and recombined code. This particular exception has never been tested in court, and the FSF's own frequently asked questions (FAQs) do not explain in detail how one would need to present a non-LGPL licensed combined work to a licensee, but at a minimum, would require providing the source code for the LGPL part of that combined work under LGPL, and at least ensuring that any facilities in the combined work—such as symbols used by symbolic debugging tools—are not removed from the object code of the application to which the LGPL part is linked.

[89] LGPLv2.1, § 5; LGPLv3, § 3.
[90] LGPLv2.1, § 6; LGPLv3, § 4.

3.2.2.1.3 The GNU Affero General Public Licence (AGPL)

The GNU Affero General Public Licence (AGPL) is the third licence in the GPL family of licences maintained by the FSF. Its intent is to close the 'ASP (application service provider) loophole' in GPL: namely, that because GPL's obligations only apply to code that is either distributed (GPLv2) or conveyed (GPLv3), entities offering Software as a Service ('SaaS') under either GPLv2 or GPLv3, such that the software is accessed by third parties—but only over a network such that the code is never distributed to them—are not obliged to supply source code to those third parties. AGPL closes this 'loophole' by adding an additional condition triggering an obligation to provide source:

> if you modify the Program, your modified version must prominently offer all users interacting with it remotely through a computer network (if your version supports such interaction) an opportunity to receive the Corresponding Source of your version by providing access to the Corresponding Source from a network server at no charge, through some standard or customary means of facilitating copying of software.[91]

The AGPL exists in two versions: AGPLv1,[92] which contains terms similar to GPLv2 (but for the additional condition recited earlier) and AGPLv3, which contains terms similar to GPLv3 (but for the additional condition recited earlier). A more detailed description of AGPL and its effect on cloud computing and SaaS can be found in Chapter 17.

3.2.2.1.4 Other copyleft licences: MPL and EPL

Although there are but a few other Open Source licences that attempt to create a strong copyleft effect, there are two very prominent other Open Source licences that create a weak copyleft effect: The Mozilla Public License (MPL)[93] and the Eclipse Public License (EPL).[94] Both of these licences are an effort—similar to the effort made with the Apache 2.0 license—to address some perceived ambiguities or missing features in LGPL, as well as to provide a clearer and more easy-to-use definition of the circumstances when non-copyleft code can make use of code licensed under those licences.

The MPL exists in two versions—MPLv1.1[95] and MPLv2.0.[96] Both are weak copyleft licences written by the Mozilla Foundation, with version 2.0 being an

[91] GNU Operating System, 'GNU Affero General Public License, Version 3', § 13, <https://www.gnu.org/licenses/agpl-3.0.en.html> accessed 19 November 2007.

[92] Software Package Data Exchange (SPDX), 'Affero General Public License v1.0 only' <https://spdx.org/licenses/AGPL-1.0-only.html> accessed 11 March 2020.

[93] Mozilla Foundation, 'Mozilla Public License' <https://www.mozilla.org/en-US/MPL/> accessed 11 March 2020.

[94] Eclipse Foundation, 'Eclipse Public License—v 2.0' <https://www.eclipse.org/legal/epl-2.0/> accessed 11March 2020.

[95] Mozilla Foundation, 'Mozilla Public License Version 1.1' <https://www.mozilla.org/en-US/MPL/1.1/> accessed 11 March 2020 (MPLv1.0).

[96] Mozilla Foundation, 'Mozilla Public License Version 2.0' <https://www.mozilla.org/en-US/MPL/2.0/> accessed 11 March 2020 (MPLv2.0).

updated version having improved terminology. Like the Apache 2.0 license versus the MIT and BSD licenses, MPL attempts to improve upon issues in LGPL—for example, by addressing patent rights via an express patent grant, including defensive termination conditions for that grant, and by having more robust definitions and terms. The MPL licences are also an attempt to make a much clearer, and easier to understand and use in practice, distinction between code that must be copyleft and code that need not be. This distinction is set forth in the definitions of 'Modifications' to the 'Covered Software' provided under MPL:[97]

> '"Modifications" means any of the following: any file in Source Code Form that results from an addition to, deletion from, or modification of the contents of Covered Software; or any new file in Source Code Form that contains any Covered Software.' MPL also make clear that putting together such files into a 'Larger Work' does not subject the entire result to the MPL: '"Larger Work" means a work that combines Covered Software with other material, in a separate file or files, that is not Covered Software.[98] … You may create and distribute a Larger Work under terms of Your choice, provided that You also comply with the requirements of this Licence for the Covered Software.'[99]

Although the file-based distinction between that which must be MPL and that which need not is likely to be much easier to navigate than the exceptions to copyleft in LGPL, there is one potential ambiguity that bears a cautious approach: when creating a new file, designed to not be subject to MPL but intended to work with it, MPL states that 'Covered Software' is defined as 'Source Code Form to which the initial Contributor has attached the notice in Exhibit A, the Executable Form of such Source Code Form, and Modifications of such Source Code Form, in each case including portions thereof'.[100] Note the degree of recursion between the definition of 'Covered Code' and 'Modifications', where each includes reference to the other. A file that is separate from Covered Code, but which may need to reproduce elements (such as interfaces) in order to work with that Covered Code, could be argued to thus fall within the obligation to license that file only under MPL and to produce source code for that file. The Mozilla Foundation attempts to clarify this issue in its FAQs:

> Q11: … If I use MPL-licensed code in my proprietary application, will I have to give all the source code away?
>
> No. The license requires that Modifications (as defined in Section 1.10 of the license) must be licensed under the MPL and made available to anyone to whom you distribute the Source Code. However, new files containing no MPL-licensed

[97] MPLv2.0, § 1.10.
[98] MPLv2.0, § 1.7.
[99] MPLv2.0, § 3.3.
[100] MPLv2.0, § 1.4.

> code are not Modifications, and therefore do not need to be distributed under the terms of the MPL, even if you create a Larger Work .... *This allows, for example, programs using MPL-licensed code to be statically linked to and distributed as part of a larger proprietary piece of software, which would not generally be possible under the terms of stronger copyleft licenses.*[101]

Given the statement about static linking in the MPL FAQs, it is certainly also the case that dynamic linking of these files would not cause the MPL to attach to the file so linked. The extent the FAQs would be dispositive on the issue of linking of MPL and non-MPL files has yet to be determined, although the fact that the licence steward, the Mozilla Foundation, accepts this distinction is likely to be found highly persuasive.

The EPL also exists in two versions—EPLv1.0[102] and EPLv2.0.[103] Both are weak copyleft licences written by the Eclipse Foundation, with version 2.0 being an updated version having improved terminology. Like MPL, EPL attempts to improve upon issues in LGPLv2.1: patent rights, patent defensive termination, and more robust definitions and terms. EPLv2.0 has a slight advantage over MPLv2.0 in that it removes from the terms of the licence the ambiguity around including interfacing code in order to allow non-EPL-licensed code to interoperate with EPL-licensed code:[104]

> 'Modified Works' shall mean any work in Source Code or other form that results from an addition to, deletion from, or modification of the contents of the Program, including, for purposes of clarity any new file in Source Code form that contains any contents of the Program. *Modified Works shall not include works that contain only declarations, interfaces, types, classes, structures, or files of the Program solely in each case in order to link to, bind by name, or subclass the Program or Modified Works thereof.*[105]

[101] Mozilla Foundation, 'MPL 2.0 FAQ' <https://www.mozilla.org/en-US/MPL/2.0/FAQ/> accessed 2 March 2020 (emphasis added).

[102] Eclipse Foundation, 'Eclipse Public License—v 1.0' <https://www.eclipse.org/legal/epl-v10.html> accessed 11 March 2020.

[103] Eclipse Foundation, 'Eclipse Public License—v 2.0' <https://www.eclipse.org/legal/epl-2.0/> accessed 11 March 2020 (EPLv2).

[104] The ambiguity present in MPLv2.0 also existed in EPLv1.0, which defined a 'Contribution' not covered by the licence as 'additions to the Program which: (i) are separate modules of software distributed in conjunction with the Program under their own licence agreement, *and (ii) are not derivative works of the Program*' (emphasis added). The reference back to the idea of derivative works begged the question of whether the use of interfacing code that might legally result in the creation of a derivative work would curtail the exception and make EPLv1.0 closer to a strong copyleft licence. See Katie Osborne, 'License Profile: The Eclipse Public License' (2015) 1(1) *Journal of Open Law, Technology & Society* 1, at 6 <https://jolts.world/index.php/jolts/article/view/73/211>.

[105] EPLv2.0, § 1 (emphasis added).

In this way, EPLv2.0 makes much clearer the distinction between EPL and non-EPL files and acknowledges that in order for such files to work together there may need to be reproduction of certain interfacing code from the EPL files. The definition of 'Modified Works' in EPLv2.0 thus acknowledges that such reproduction does not spread the licence to that file.

3.2.2.1.5 'Weak copyleft' exception practice

In certain circumstances, authors have chosen to license their code under a strong copyleft licence but have attempted to provide a particular weak copyleft effect to enable certain uses of the code without imposing strong copyleft requirements on that code. Two of the most notable examples of this is the so-called sys call exception that is used with the Linux kernel (part of the GNU/Linux operating system) and the 'runtime' exception used with the GNU C++ Compiler (GCC). In each case, a version of GPL is used for the base code licence, but the authors have included a notice (usually included immediately above the licence text, or in a 'README' file appended to the source code) to spell out that for certain uses, GPL would not apply.

The Linux kernel, licensed under GPLv2, includes the following exception (often called the 'sys call' or 'system call' exception), authored by the original author of that kernel, Linus Torvalds:

> NOTE! This copyright does *not* cover user programs that use kernel services by normal system calls—this is merely considered normal use of the kernel, and does *not* fall under the heading of 'derived work'. Also note that the GPL below is copyrighted by the Free Software Foundation, but the instance of code that it refers to (the Linux kernel) is copyrighted by me and others who actually wrote it.[106]

In this way, the authors of the Linux kernel attempt to weaken some of the effect of GPLv2 against their authored code by making clear that as authors, and therefore granters of the licence, certain uses that other code might make of facilities in the kernel, even if in the law of a relevant jurisdiction might be considered a derivative work subject to the requirements of GPLv2, are not considered so by those authors. In this way, for this particular code, GPLv2 is turned into a weak copyleft licence, but not using the definitions of exceptions to the licence that are used in, *inter alia*, LGPL.

The GCC also uses an exception to specify that certain uses of the code from that tool—licensed under GPLv3—do not spread the licence attached to that code.[107]

[106] <https://github.com/torvalds/linux/blob/master/LICENSES/exceptions/Linux-syscall-note> accessed 13 April 2022.

[107] GNU Operating System, 'GCC Runtime Library Exception version 3.1' (31 March 2009) <https://www.gnu.org/licenses/gcc-exception-3.1.en.html> accessed 13 April 2022.

The wording of the exception itself is somewhat complex, but the intent of the exception is described before the text of that exception:

> When you use GCC to compile a program, GCC may combine portions of certain GCC header files and runtime libraries with the compiled program. The purpose of this Exception is to allow compilation of non-GPL (including proprietary) programs to use, in this way, the header files and runtime libraries covered by this Exception.[108]

The purpose of this exception is to prevent mere use of the tool to compile code from causing the resulting compiled code to be GPL—thus limiting the use of the tool itself to only GPLv3 licensed programs.[109]

There are numerous other examples of software licensed under an Open Source licence where the authors have granted some form of exception where certain uses of their code are not obliged to comply with all or part of the terms of that licence; it is good practice, when confronted with situations where a proposed use of Open Source licensed software might result in a less than desirable licensing outcome (either a licence conflict, or the requirement to use a licence that may be undesirable in a particular usage case) to examine the source code repository to see if an author exception has been granted.

3.3 Software Interaction and Licence Compatibility

3.3.1 The linking question

As described in section 3.1.4.1, linking—either statically, before run-time, or dynamically, during run-time—is a relatively common programing technique to allow two or more software programs or modules to operate together or to be combined. There has long been a debate about the effect of linking on programs licensed using Open Source licences—particularly copyleft licences.[110] Some licences, like Apache, attempt to make clear that irrespective of whether the law in a particular jurisdiction would find a particular type of link to create a derivative work, certain forms of linking do not cause the licence to apply to the result. In that case, it is relatively safe to assume that such an exception would be found controlling on the question of applicability of the licence. Other licences, like GPLv3, attempt to make clear that certain types of linking should be considered to create a derivative

[108] 'GCC Runtime Library Exception version 3.1' see note 111.

[109] GNU Operating System, 'GCC Runtime Library Exception Rationale and FAQ' <https://www.gnu.org/licenses/gcc-exception-3.1-faq.html> accessed 11 March 2020.

[110] Bain, 'Software Interactions and the GNU General Public License', see note 83, at 177.

work (or at least, to be governed by the terms of the licence) and therefore cause the licence to apply to the result. Whether the courts in a particular jurisdiction would find such a statement of intent applicable is an as-yet unresolved issue.

There is also an important issue concerning the particular jurisdiction in which the linking question is confronted. The term 'derivative work' has a statutory meaning under US law, although it is subject to different interpretive tests at the present, as discussed in more detail in section 3.1.4.2 earlier in this chapter. In other jurisdictions such as those in the UK and EU, the analysis is not even that clear-cut, due to the lack of definition of the term.

3.3.1.1 Other interaction issues: technical impediments

The Linux kernel, licensed under GPLv2 only, instituted—in around 2002—a technical impediment intended to discourage the loading of LKMs at run-time which were not licensed under GPLv2 or a GPLv2 compatible licence.[111] This technical impediment issues an error message ('Kernel is Tainted for following reasons: Proprietary module was loaded') when such a non-GPLv2 compatibly licensed LKM is loaded by the kernel.[112] In general, the community of Linux kernel developers and maintainers believe that run-time LKMs that are 'proprietary' (typically closed source, but this impediment would also display a message for LKMs licensed under a non-GPLv2-compatible licence) should not be allowed, and this technical impediment was intended to discourage the creation or use of such LKMs.[113] There have been efforts to circumvent this technical impediment in order to allow the loading of proprietary LKMs without the display of the warning message, although the reaction by the kernel maintainer community to such efforts is decidedly negative if not hostile.[114] Avoiding such impediments to allow proprietary LKMs to load may also potentially run afoul of laws—like the DMCA in the US—intended to prevent circumvention of technological impediments by way of DRM.

3.3.2 Specific compatibility issues

Navigating which of the many Open Source licences are compatible with one another, and under what technical conditions and in which legal jurisdictions, is a highly complex issue. The answer will depend at least in part on how two or more pieces of software interact with one another, the extent to which interfaces or other

[111] Linux Kernel Mailing List (LKML), 'The tainted message' (26 April 2002), <http://lkml.iu.edu/hypermail/linux/kernel/0204.3/0428.html> accessed 13 April 2022.

[112] Linux Kernel, 'The Linux kernel user's and administrator's guide: Tainted kernels', <https://www.kernel.org/doc/html/latest/admin-guide/tainted-kernels.html> accessed 11 March 2020.

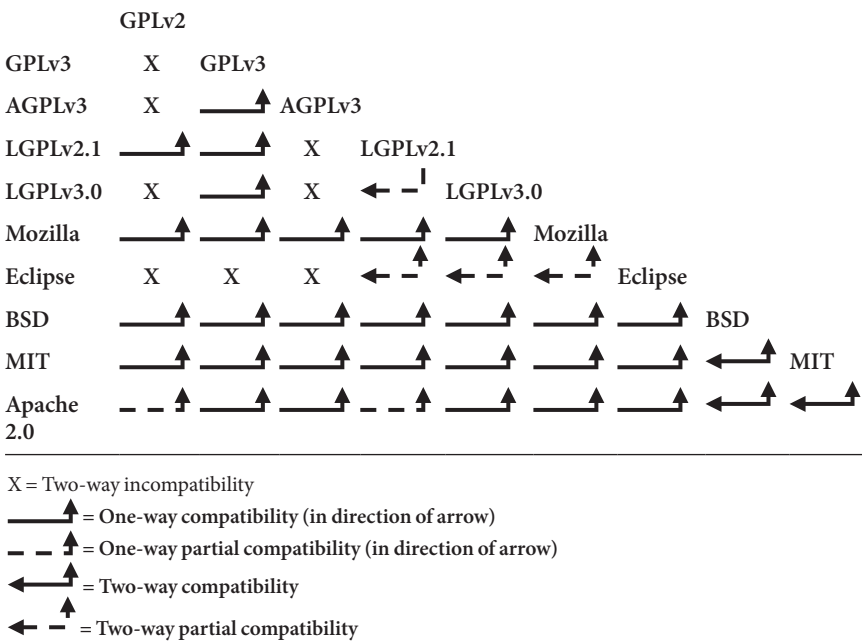
[113] Linux.com, 'Tainted love: proprietary drivers and the Linux kernel' (28 April 2004), <https://www.linux.com/news/tainted-love-proprietary-drivers-and-linux-kernel/> accessed 13 April 2022.

[114] LKML, 'Linuxant/Conexant HSF/HCF modem drivers unlocked' (29 October 2004) <http://lkml.iu.edu/hypermail/linux/kernel/0410.3/2190.html> accessed 13 April 2022.

code may need to be reproduced to facilitate that interaction, the specific wording of the licences involved, and whether there are any exceptions granted by the authors that might contemplate and except out from general licence conditions certain types of interactions. The discussion which follows attempts to summarise these compatibility issues, but practitioners are cautioned that there are many different factors that need to be considered, and the law on, *inter alia*, copyrightability, functional dictation/merger doctrine, and fair dealing/fair use continue to evolve, and cases still being considered as of publication may significantly impact questions related to compatibility.

Table 3.2 is a graphical representation intended to summarise how the licences discussed earlier are, are not, or may possibly be, compatible. A single-headed arrow represents 'one way' compatibility—that is the licence at the start of the arrow is compatible with the licence at the end of the arrow, but not vice versa; and a double-headed arrow represents 'two way' compatibility, that is the licences are compatible in either direction. A solid line means compatibility in all circumstances, a dashed line means compatibility in certain circumstances, and a dotted line means there is an unresolved debate about whether compatibility exists. An 'X' means there is no compatibility in either direction. Note that this chart does not take into account that there may be some degree of compatibility in certain specific

Table 3.2 Compatibility between Certain Open Source Licences



circumstances as a result of author exceptions, as discussed in section 3.2.2.1.5 earlier, or licence statements that allow the user to receive the code under later versions of the licence, as discussed in section 3.3.2.1 which follows.

3.3.2.1 'Strong' copyleft licences

Those licences which are generally referred to as having 'strong' copyleft provisions have the hallmark of imposing their terms on any downstream exercise of the right to make modifications (or in the case of the US, create 'derivative works'). Because of this, attempting to interoperate 'strong' copyleft licensed code with code that has any terms that might conflict with the terms in the 'strong' copyleft licence, presents potential compatibility problems that puts the interoperating code at risk of violating the 'strong' copyleft licence's terms (as well as the terms for the interoperating code). In this way, the strong copyleft licences (GPLv2, GPLv3, AGPL) are generally incompatible with one another. There are a few exceptions to this general rule. First, there has been a practice for a number of authors of code licensed under GPLv2 to include a statement that the code is licensed under 'GPLv2 or any later version'.[115] Code licensed under this form of a GPLv2 licence notice is compatible with GPLv3; otherwise, code licensed under GPLv2 only (which is the case with the Linux kernel)[116] is incompatible with GPLv3, LGPLv3, and AGPLv3.[117] Second, the newer versions of the GNU family of licences were designed to allow a certain degree of one-way compatibility with one another; thus, LGPLv3 may be combined with GPLv3 as long as the resulting combination is licensed GPLv3;[118] similarly, AGPLv3 includes a provision that allows combinations with GPLv3, as long as the resulting combination is licensed GPLv3.[119] Third, LGPLv2.1 includes a section stating that 'You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy' of code licensed under LGPLv2.1;[120] thus LGPLv2.1 is one-way compatible with GPLv2 and GPLv3.

3.3.2.2 'Weak' copyleft licences

The weak copyleft licences tend to have limited compatibility. A weak copyleft licence typically stipulates that derivative content must be licensed under that same licence, but admits of articulated exceptions to that general rule.

[115] <https://spdx.org/licenses/GPL-2.0-or-later.html> accessed 13 April 2022.

[116] Kernel.org, 'Working with the kernel development community: Linux kernel licensing rules' <https://www.kernel.org/doc/html/latest/process/license-rules.html> accessed 11 March 2020.

[117] Free Software Foundation, 'Frequently Asked Questions about the GNU Licenses' <https://www.gnu.org/licenses/gpl-faq.en.html#AllCompatibility> accessed 3 March 2020 (the chart does not address AGPLv3, but by the logic of GPLv3 incompatibility, AGPLv3 would be equally incompatible).

[118] FSF, 'Frequently Asked Questions about the GNU Licenses' see note 121.

[119] AGPLv3, § 13.

[120] LGPLv2.1, § 3.

The MPL includes provisions, Sections 1.12 and 3.3, designed to allow combinations with the GNU family of licences, such that combinations of the two would be governed by the terms of the GNU-family licence. Thus, MPL is one-way compatible with GPLv2.1, GPLv3, LGPLv2.1, LGPLv3, and AGPLv3.[121] With regard to the EPL, depending on which version of MPL and EPL, and the way in which code under MPL and EPL are designed to interoperate, there is the potential for the licences to be compatible; this is primarily an issue of whether the code is maintained in separate files, as discussed in more detail in section 3.2.2.1.4 earlier in this chapter.

The EPL, in contrast, does not include a provision designed to allow compatibility with the GNU family of licences; thus the EPL is incompatible with all of GPLv2.1, GPLv3, and AGPLv3.[122] With respect to LGPLv2.1 and LGPLv3, it is possible to construct an interaction between code under EPL and one of the LGPL licences, as long as doing so falls within the exceptions in both licences discussed in more detailed in sections 3.2.2.1.2 and 3.2.2.1.4 earlier.

Finally, LGPLv2.1 would normally be considered incompatible with LGPLv3, for many of the same reasons that GPLv2 is considered incompatible with GPLv3. However, LGPLv2.1 includes a provision that allows re-licensing of LGPLv2.1 code under GPLv2 or any later version of that licence.[123] LGPLv3 includes a similar provision.[124] Thus, LGPLv2.1 and LGPLv3 can be made compatible, but the result would be licensing under GPLv3.[125]

3.3.2.3 MIT and BSD

Few complications arise with respect to compatibility between and with the permissive licences; indeed, it is a feature of these licences to provide broad compatibility. Copying and linking (which would appear to fall within the broad grant of rights in these licences) are broadly permitted by MIT and BSD, with only minimal requirements. Thus, as shown in Table 3.2, both of these licences are one-way compatible with all the Open Source licences discussed above, and are two-way compatible with each other and with Apache.

3.3.2.4 Apache

The Apache License also has a goal of being highly permissive and broadly compatible, in the same way as MIT and BSD. In most instances, as shown in Table 2 above,

[121] Free Software Foundation, 'Various licenses and comments about them' <https://www.gnu.org/licenses/license-list.html#MPL-2.0> accessed 11 March 2020.

[122] Free Software Foundation, 'Various licenses and comments about them' <https://www.gnu.org/licenses/license-list.html#EPL> accessed 11 March 2020.

[123] LGPLv2.1, § 3.

[124] LGPLv3, § 2b.

[125] Free Software Foundation, 'Frequently Asked Questions about the GNU licenses' <https://www.gnu.org/licenses/gpl-faq.en.html#AllCompatibility> accessed 11 March 2020.

the Apache License achieves that goal—being one-way compatible with most of the licences discussed above, and two-way compatible with MIT and BSD.

The Apache License does present a complication with regard to the GNU family of licences. The FSF has stated that the Apache 2.0 license is incompatible with GPLv2.[126]Although the ASF disagrees with this assessment, it nevertheless states that with regard to use of GPLv2, 'you should always try to obey the constraints expressed by the copyright holder when redistributing their work'.[127] This incompatibility introduces certain complications in architecting software stacks that may include Apache 2.0 and GPLv2 code (in particular, the Linux kernel of the GNU/Linux operating system, which is licensed under GPLv2 only), and could very well be the reason why the more recent, and more carefully drafted, Apache 2.0 license has not supplanted the continued popularity of the BSD and MIT licences as *de facto* choice when selecting a permissive licence. Note that this incompatibility does not exist for the later generations of the GNU family of licences—GPLv3, LGPLv3, and AGPLv3—as it was an express goal during the process of updating those licences to allow them to be Apache License one-way compatible.[128] Thus, as reflected in Table 3.2, the Apache License is arguably one-way compatible with GPLv2, partially one-way compatible with LGPLv2.1, and fully one-way compatible with GPLv3, LGPLv3, and AGPLv3.

3.4 Interpreting Open Source Licences: Contract or 'Bare Licence'?

Although a discussion of specific enforcement cases and issues are described and analysed in detail in Chapter 5, there has been a long-standing debate amongst Open Source licence drafters, users, and potential enforcers over the question of whether such licences operate as 'bare licences', or should be interpreted and enforced as contracts. This question can be thought of—until very recently—as significantly academic, and much of the debate has been in academic circles but is nonetheless important when a particular author is (i) choosing a licence for their work, or (ii) considering enforcing that licence against another person or entity which they believe to be failing to comply with its terms. Although there is as yet no definitive answer to this debate—and in fact, the answer may be dependent upon the particular licence being enforced, and possibly the jurisdiction in which

[126] See FSF Licence Comments, see note 59, <https://www.gnu.org/licenses/license-list.html#apache2>. Note that by the logic used in the FSF's commentary on the Apache 2.0 licence, Apache 2.0 would likely be also incompatible with the Affero GPLv1.0 licence, and partially incompatible with the Lesser GPLv2.1 licence.

[127] Apache Software Foundation, 'GPL compatibility' <https://www.apache.org/licenses/GPL-compatibility.html> accessed 10 March 2020.

[128] Brett Smith, 'A Quick Guide to GPLv3' <https://www.gnu.org/licenses/quick-guide-gplv3.html> accessed 11 March 2020.

enforcement is contemplated—the trend in adjudication would appear to be supportive of the bare licence theory of licence interpretation and enforcement, or indeed possibly that either theory could be pursued by an author of software licensed under an Open Source licence, in the event they desire to engage in licence enforcement.

3.4.1 Open Source licences as bare licences

It has long been the position of the FSF that the licences over which it exercises stewardship operate as 'bare licences'.[129] Under this theory, the author grants a unilateral permission, under the IP rights either expressly enumerated in the licence text, or impliedly granted as a result of the structure and text of the licence and conditions under which it was granted, to engage in activities the author would otherwise have exclusive rights to practice.

Thinking of Open Source licences—or at least those licences that do not explicitly present themselves as contracts between the author(s) and licensee(s)—as mere unilateral permission from the author to each particular user, provides some potentially advantageous benefits to the author(s). First, the necessity to worry about the fundamental requirements for establishing the existence of a contract—offer, acceptance, consideration, intent, certainty, and completeness[130]—need not be established in order to pursue a violator, nor may privity of contract with the particular violator need be established. Second, given the numerous variations in rules governing contract law (which, for example, are analysed state-by-state in the US), viewing an Open Source licence as merely a permission under certain IP rights which—if not followed—result in a claim for violation of those rights, may allow the application of more uniform law, and provide more flexibility and ease in pursuing remedies, than pursuing relief under contract.[131]

The trend of enforcement and judicial interpretation of Open Source software licences suggests that the bare licence theory of Open Source licence interpretation and enforcement is valid, and may be preferable to those authors wishing to exercise their right of enforcement against accused licence violators.

The *Jacobsen v Katzer*[132] case in the US led the way—at least in common law jurisdictions—in validating the bare licence theory of Open Source software

[129] See Eben Moglen and Richard Stallman, 'Transcript of Opening Session of the First International GPLv3 Conference', 16 January 2006 <http://www.ifso.ie/documents/gplv3-launch-2006-01-16.html> accessed 16 January 2020. Eben Moglen, 'Enforcing the GNU GPL' (10 September 2001), <https://www.gnu.org/philosophy/enforcing-gpl.html> accessed 28 February 2020.

[130] Catharine MacMillan and Richard Stone, 'Elements of the law of contract' (2012) University of London International Programmes, <https://www.dphu.org/uploads/attachements/books/books_4071_0.pdf> accessed 13 April 2022.

[131] See GPLv3 Transcript, note 28.

[132] *Jacobsen v Katzer,* 535 F.3d 1373 (Fed. Cir. 2008).

licensing. The *Jacobsen* case involved enforcement of the terms of the Artistic Licence,[133] a permissive licence that includes obligations to include attribution notices and to identify modifications made. The defendant used code licensed by the plaintiff under the Artistic License, but failed to provide attribution or identify modifications. According to the District Court in *Jacobsen*, defendant's violation of the requirements of the Artistic License constituted a breach of contract, rather than use of the plaintiff's copyrights outside of the conditions of the licence and thus copyright infringement.[134] The Court of Appeals for the Federal Circuit overruled the District Court's ruling, holding that the requirements of Artistic License were not independent contractual covenants but merely conditions attached to the copyright grant. Because the defendant's actions had gone beyond the scope of the licence—by failing to comply with the fairly minimal conditions of the Artistic License—an action for copyright infringement could be brought by the author, and remedies for copyright infringement could be sought.[135]

Given the similarity in the grants and conditions between the Artistic License and the BSD and MIT Licenses, it would seem likely that at least those licences would also be interpreted to operate as bare licences, at least in the US, under the reasoning of the *Jacobsen v Katzer* decision. As noted earlier, the FSF—stewards of the GNU family of licences—has long advocated that those licences are also bare licences and not contracts, and commentators have acknowledged that that may be a viable interpretation of those licences.[136] Although there is no clear UK case regarding the bare licence versus contract issue concerning Open Source licensing, some commentators believe the rationale of *Jacbosen v Katzer* would equally apply there.[137]

Civil law jurisdictions also appear to generally accept the bare licence theory as at least one way to interpret Open Source licences.[138] A recent decision of the ECJ, upon appeal of a decision emanating from France, seems to bear out the theory

[133] Ironically, given that this case is perhaps the most consequential decisions interpreting the obligations of an Open Source licence, it involves one of the least popular Open Source licences. *See* Ben Balter, 'Open Source License Usage of GitHub.com' *The GitHub Blog* (9 March 2015), <https://github.blog/2015-03-09-open-source-license-usage-on-github-com/> accessed 3 February 2021 (showing the Artistic Licence as the seventh most used of sixteen open source licences on GitHub).

[134] *Jacobsen v Katzer*, No. 06-CV-01905 JSW, 2007 WL 2358628 (N.D.Cal. 17 August 2007).

[135] *Jacobsen v Katzer*, 535 F.3d 1373 at 1381–3 (Fed. Cir. 2008).

[136] Mark Henley, 'Jacobsen v Katzer and Kamind Associates—An English Legal Perspective' (2009) 1(1) *Journal of Open Law, Technology and Society* 41, at 43 (2009); Noah Shemtov, 'FOSS License: Bare License or Contract', presentation available at <https://web.ua.es/es/contratos-id/documentos/itipupdate2011/shemtov.pdf> accessed 12 March 2020.

[137] Shemtov, 'FOSS License: Bare License or Contract', see note 140.

[138] German cases include *Welte v Sitecom Deutschland GmbH*, District Court of Munich, 19 May 2004, case 21 O 6123/04; *Welte v Skype Technologies S A*, District Court of Munich, 12 July 2007, case 7 O 5245/07. A French case, *EDU 4 v AFPA*, Cour d'Appel de Paris, Pole 5, Chambre 10, no: 294, discusses GPL although not in detail. See Martin von Willebrand, 'Case Law Report: A Look at EDU 4 v. AFPA, also Known as the "Paris GPL case"' (2009) 1(2) *Journal of Open Law, Technology and Society* 123, at 123–26.

that a copyright licence violation can indeed be pursued whether or not there may be a contractual basis for a claim against the licence violator:

> According to Article 2(1) of Directive 2004/48 [of the European Parliament and of the Council of 29 April 2004 on the enforcement of intellectual property rights], that directive applies to 'any infringement of intellectual property rights.' It is apparent from the wording of that provision, in particular from the adjective 'any', that that directive must be interpreted as also covering infringements resulting from the breach of a contractual clause relating to the exploitation of an intellectual property right, including that of an author of a computer program.[139]

3.4.2 Open Source licences as contracts

There is a relatively robust line of argument that Open Source licences—or at least a selected subset of Open Source licences—operate as valid contracts between the authors of the licensed code and the recipients of that code.[140] Others have argued to the contrary:

> A contract … is an exchange of obligations, either of promises for promises, or of promises of future performance, for present performance, or payment. The idea that 'licenses' to use patents or copyrights must be contracts is an artefact of twentieth-century practice, in which licensors offered an exchange of promises with users: 'We will give you a copy of our copyrighted work,' in essence, 'if you pay us and promise to enter into certain obligations concerning the work.'[141]

As discussed earlier, several of the more commonly-used Open Source licences are likely to be evaluated as bare licences—if the author chose to present that theory to a judicial tribunal—although the interpretive decisions validating that theory do not preclude an author from also pursuing a claim for breach of contract.

At least one court has interpreted GPLv2 under a contractual analysis and has rejected the application of a 'bare licence' theory for enforcement of that licence. In the French court decision in *Entre'Ouvert v Orange*,[142] the court—in

[139] *IT Development SAS v Free Mobile SAS*, ECLI:EU:C:2019:1099 (Fifth Chamber, CJEU, 18 December 2019). Compare that decision to the outcome in *Entre'Overt v Orange*, Tribunal de grande instance (TGI) of Paris, 3rd chamber, 3rd section (21 June 2019), discussed in section 3.4.2 below.

[140] See Robert W Gomulkiewicz, 'How Copyleft Uses License Rights to Succeed in the Open Source Software Revolution and the Implications for Article 2B' (1999) 36 *Houston Law Review* 179, at 194; Rosen, 'Open Source Licensing', see note 67, at 57–66.

[141] Eben Moglen, quoted in Pamela Jones, 'The GPL is a License, Not a Contract' lwn.net (3 December 2003) <https://lwn.net/Articles/61292/> accessed 2 March 2020.

[142] *Entre'Ouvert v Orange*, Tribunal de grande instance (TGI) of Paris, 3rd chamber, 3rd section (21 June 2019) <https://www.legalis.net/jurisprudences/tgi-de-paris-3eme-ch-3eme-section-jugement-du-21-juin-2019/> accessed 8 June 2022.

interpreting a claim of failure to follow the requirements of GPLv2—stated that because Entre'Ouvert was seeking compensation for damage caused by Orange's failure to perform obligations in GPLv2—specifically providing source code—the defendant was not operating outside of the GPLv2 license and as a result the only claim Entre'Ouvert could pursue was under French contract law, for a contractual breach of the requirements of GPLv2.[143] This decision would appear to be conflict with the ECJ's interpretation of French law in *IT Development SAS v Free Mobile SAS*,[144] despite the *Entre'Ouvert* decision being issued previously, but which was not cited in the *IT Development* decision. The extent to which this conflict will be resolved, and its effect on potential enforcement actions in France, have not yet been clearly established.

In 2021, a lawsuit was filed in the US with the intent of definitively establishing not only a contract theory for the GPL family of licences, but also to open up the possibility that recipients of Open Source—rather than just authors of Open Source—could enforce the terms of those licences. In *Software Freedom Conservancy, Inc. v Vizio, Inc.*,[145] a lawsuit was filed—in the state courts of California—to enforce GPLv2 based on allegations that 'complete corresponding source' had not been provided to purchasers of products sold by Vizio containing GPLv2 binaries.[146] The Software Freedom Conservancy, as one of the purchasers, asserted it was a third-party beneficiary of the contractual right in GPLv2 to receive source code, and therefore had the right to sue to enforce that licence.[147] In response, Vizio attempted to have that lawsuit 'removed' (transferred) from state court to US federal court, arguing that any violation of GPLv2 may only be pursued as a claim of copyright infringement, which are heard exclusively in US federal courts.[148] The federal court decided that violations of the obligation to provide source under GPLv2 could be pursued as a matter of contract law, and therefore US state courts could decide such claims under state contract law.[149] Thus, at least in the US, the potential for pursuing GPL violation claims as copyright infringements, in US federal courts, and as contract breaches, in US state courts, may be a possibility—depending on the eventual outcome of the *Vizio* litigation.

At least some Open Source licences intentionally present themselves as contracts as well as licences.[150] At a minimum, the most popular Open Source licences

[143] *Entre'Ouvert v Orange*, note 146.

[144] *IT Development SAS v Free Mobile SAS*, ECLI:EU:C:2019:1099 (Fifth Chamber, CJEU).

[145] Case No. 30-2021-01226723-CU-BC-CJC (Cal. Super. Ct., Orange County, filed 19 October 2021) (*Vizio* state case).

[146] *Vizio* state case, note 149, Complaint at paras 48–77.

[147] *Vizio* state case, note 149, Complaint at paras 87–126.

[148] *Software Freedom Conservancy, Inc. v Vizio, Inc.* Case No. 8:21-cv-01943, Notice of Removal of Action to Federal Court (C. D. Cal. 29 November 2021) (*Vizio* federal case).

[149] *Vizio* federal case, note 152, Order Granting Plaintiff's Motion for Remand (C. D. Cal. 13 May 2022).

[150] Rosen, 'Open Source Licensing', see note 67, at 59 (discussing the Academic Free License and the Open Software License).

do not explicitly preclude an interpretation that they could be enforced as bare licences, but only as contracts. Given the weight of both enforcement theories, and judicial decisions to date, it seems clear that the contract theory could remain a minority theory of Open Source software licensing interpretation, and is unlikely to a majority theory of license enforcement absent a significant decision calling into question the application of the bare licence theory to particular licences or in particular jurisdictions.

3.5 What Makes a Software Licence 'free' or 'open source'?

Although this volume refers to 'free and open source licences collectively as open source, that use does not necessarily represent a unitary concept. 'Free and open source' licensing actually represent two different, but significantly coextensive, classes of licences: 'free' software licences, and 'open source' software licences.

3.5.1 Free software licences

The class of 'free' software licences is typically recognised as those licences that meet the Free Software Definition (FSD),[151] as maintained by the FSF, and that have been validated by the FSF and added to their list of free software licences.[152] The FSD is a four-part test against which licences are measured to determine if they promote the FSF's concept of 'software freedom':

- The freedom to run the program as you wish, for any purpose (freedom 0).
- The freedom to study how the program works, and change it so it does your computing as you wish (freedom 1). Access to the source code is a precondition for this.
- The freedom to redistribute copies so you can help others (freedom 2).
- The freedom to distribute copies of your modified versions to others (freedom 3). By doing this you can give the whole community a chance to benefit from your changes. Access to the source code is a precondition for this.[153]

The FSF's 'four freedoms' are the minimum standards necessary for a particular software licence to be considered a 'free software' licence; as the FSF states:

[151] GNU Operating System, 'What is free software? The Free Software Definition' <https://www.gnu.org/philosophy/free-sw.html.en> accessed 4 February 2021.

[152] GNU Operating System, 'Various licenses and comments about them: software licenses' <https://www.gnu.org/licenses/license-list.html#SoftwareLicenses> accessed 4 February 2021.

[153] FSD, see note 146.

> [C]riteria such as those stated in this free software definition require careful thought for their interpretation. To decide whether a specific software license qualifies as a free software license, we judge it based on these criteria to determine whether it fits their spirit as well as the precise words. If a license includes unconscionable restrictions, we reject it, even if we did not anticipate the issue in these criteria. Sometimes a license requirement raises an issue that calls for extensive thought, including discussions with a lawyer, before we can decide if the requirement is acceptable. When we reach a conclusion about a new issue, we often update these criteria to make it easier to see why certain licenses do or don't qualify.[154]

The FSF maintains an extensive list of software licences that it has determined, based on the criteria discussed earlier, meet its standards in order to qualify as a 'free software' licence, as well as a list of licences which it has determined are 'non-free' because they fail these criteria.[155] The 'free software' licences are subdivided in two categories: 'free software' licence that are 'GPL-compatible' and those that are 'GPL-incompatible'.[156] The measure of GPL compatibility is determined by evaluating whether the licence in question is one-way compatible with GPL.[157] GPL compatibility is an important criterion for the FSF, as the FSF promotes the GPL as the optimal licence for software freedom.[158]

There is no formal process for validating that a licence is a 'free software' licence and therefore to add a licence to the FSF's list of 'free software' licence; licences may be submitted via email to the FSF, but there is no formal review process or timeline specified by the FSF for making such decisions or adding licences to its lists.[159]

3.5.2 Open source software licences

The class of 'open source' software licences is recognised as those licences that meet the Open Source Definition (OSD), as maintained by the OSI, and that have been validated by an approval process run by the OSI. The OSD is a ten-part test against which licences are measured to determine if are 'open source':

1. Free Redistribution
   The license shall not restrict any party from selling or giving away the software as a component of an aggregate software distribution containing

[154] FSD, see note 146.
[155] FSF Software License List, see note 147.
[156] FSF Software License List, see note 147.
[157] GNU Project, 'What does it mean to say a license is "compatible with the GPL?"' <https://www.gnu.org/licenses/gpl-faq.html#WhatDoesCompatMean> accessed 5 February 2021.
[158] GNU Project, 'Why you shouldn't use the Lesser GPL for your next library' <https://www.gnu.org/licenses/why-not-lgpl.html> accessed 5 February 2021.
[159] FSF Software License List, see note 147.

programs from several different sources. The license shall not require a royalty or other fee for such sale.

2. Source Code
   The program must include source code, and must allow distribution in source code as well as compiled form. Where some form of a product is not distributed with source code, there must be a well-publicized means of obtaining the source code for no more than a reasonable reproduction cost, preferably downloading via the Internet without charge. The source code must be the preferred form in which a programmer would modify the program. Deliberately obfuscated source code is not allowed. Intermediate forms such as the output of a preprocessor or translator are not allowed.
3. Derived Works
   The license must allow modifications and derived works, and must allow them to be distributed under the same terms as the license of the original software.
4. Integrity of The Author's Source Code
   … The license must explicitly permit distribution of software built from modified source code.…
5. No Discrimination Against Persons or Groups
   The license must not discriminate against any person or group of persons.
6. No Discrimination Against Fields of Endeavor
   The license must not restrict anyone from making use of the program in a specific field of endeavor.…
7. Distribution of License
   The rights attached to the program must apply to all to whom the program is redistributed without the need for execution of an additional license by those parties.
8. License Must Not Be Specific to a Product
   The rights attached to the program must not depend on the program's being part of a particular software distribution.…
9. License Must Not Restrict Other Software
   The license must not place restrictions on other software that is distributed along with the licensed software.…
10. License Must Be Technology-Neutral
    No provision of the license may be predicated on any individual technology or style of interface.[160]

The OSI maintains an extensive list of software licences that it has determined, based on the criteria discussed earlier, meet its standards in order to qualify as an

[160] Open Source Initiative, 'The Open Source Definition' <https://opensource.org/osd> accessed 5 February 2021.

'open source' licence.[161] The list is sorted into one sub categorisation, established by the OSI, so as to identify licences that are 'are popular, widely used, or have strong communities', as well as other subcategories of licences that do not meet that test.[162] The 'popular, widely used, or have strong communities' licences as identified by the OSI are:

- Apache License 2.0
- BSD 3-Clause and BSD 2-Clause Licenses
- All versions of GPL
- All versions of LGPL
- MIT License
- Mozilla Public License 2.0
- Common Development and Distribution License (CDDL)
- Eclipse Public License version 2.0

Unlike the 'free software' licence list maintained by the FSF, the OSI has a formal, documented process for submitting, evaluating, and approving licences to add to the list of 'open source' licences.[163] Licence submitters are requested to provide the following information concerning the licence for which they request OSI approval:

- Rationale: Clearly state rationale for a new license
- Distinguish: Compare to and contrast with the most similar OSI-approved license(s)
- Legal review: Describe any legal review the license has been through, and provide results of any legal analysis if available
- Proliferation category: Recommend which license proliferation category is appropriate[164]

The process itself is administered using a mailing list through which OSI members may submit comments, criticisms, or suggested changes regarding submitted licences, after which the OSI board conducts a vote as to whether a submitted licence should be added to the OSI 'open source' licence list, or other alternative actions are taken:

[161] Open Source Initiative, '[Open Source] Licenses by Name' <https://opensource.org/licenses/alphabetical> accessed 5 February 2021.

[162] Open Source Initiative, '[Open Source] Licenses by Category' <https://opensource.org/licenses> accessed 5 February 2021.

[163] Open Source Initiative, 'The License Review Process' <https://opensource.org/approval> accessed 5 February 2021.

[164] OSI License Review Process, see note 158.

- Defer for another 30-day discussion cycle, if community discussion of conformance of the license to the OSD remains active.
- Approve if, after taking into consideration community discussion, the OSI determines that the license conforms to the OSD and guarantees software freedom. A license may be approved on the condition that a change be made, but in general a license requiring changes will have to be resubmitted.
- Reject if (a) the OSI determines that the license cannot practically be remedied to adequately guarantee software freedom, or (b) there is sufficient consensus emerging from community discussion that the license should be rejected for substantive reasons, or (c) the license is problematic for non-substantive reasons (for example, it is poorly drafted or significantly duplicative of one or more existing OSI-approved licenses).
- Withhold approval, if (a) the OSI determines that approval would require reworking the license and (b) the license submitter appears willing and able to revise the license constructively.[165]

Although the OSI licence review process is open and relatively transparent, it has not been without controversy in some circumstances, and submitters have withdrawn licences from approval that they maintain are conformant with the OSD but which have encountered opposition during the approval process.[166]

3.6 Conclusion

Although the applicability of copyright to software—in any form that it may take—is by now well-established, and despite the over thirty-year history of Open Source licensing, there remain many unresolved questions about how software copyright should be analysed legally, or how certain aspects of Open Source software licences would be found to operate, if put to the test via court or other challenges.

Chapter 5 addresses the enforcement cases that have been pursued, and this chapter gives an overview of that which is—at the present—known about the operation of Open Source licences alone, or together with other licences.

Practitioners attempting to give advice on complex questions of licence interpretation in view of particular software programing and architectural scenarios should be cautious to appreciate that many questions remain unanswered, and the views expressed by many commentators can diverge, sometime quite radically. In addition, the answer to certain thorny questions about, for example, under

[165] OSI License Review Process, see note 158.

[166] Elliot Horowitz, 'Approval: Server Side Public License, Version 2 (SSPL v2)' (9 March 2019) <http://lists.opensource.org/pipermail/license-review_lists.opensource.org/2019-March/003989.html> accessed 5 February 2021.

what conditions a copyleft licence imposes its obligations on other software may be highly dependent upon the exact wording of that licence, the general state of copyright law interpretation both internationally, and nationally in the particular jurisdiction in which the software is being used, and the extent to which a deciding or interpreting body views the terms of that licence as the mere grant of enumerated copyright rights subject to certain conditions, or a reciprocal licence between the authors of the copyleft licensed code and the authors of the other software code. There are a handful of decided cases, in both the US and EU, which give some general guidance about how these issues might be resolved, but there is much room for additional decisions which could very well upend the way that Open Source licences are interpreted, and used in practice, today.