An introduction to law for computer scientists.

Dr David Ham

May 5, 2020

Lecture 1

Principles of copyright in software

1.1 What is copyright?

Copyright is a set of *legal property rights* which the author of a work, or their employer, is granted by statute. These allow the copyright owner to control certain acts in relation to that work. The original intention of copyright law is to allow owners to generate income from their works by acts such as selling copies. As we shall see, the rights granted by copyright statutes can also be used to control the use of works in other ways.

1.2 The legal basis of copyright

Copyright is a statutory right created in UK domestic law by legislation passed by parliament. The current copyright act in the UK is the Copyright Designs and Patents Act 1988. However copyright exists in a wider context of European and international law. This creates a system of copyright law which is much more internationally uniform than is the case for other domains of law such as contract, tort[1] and criminal law. This means that most of what we will cover in these lectures will hold true in many other jurisdictions. Attempts will be made to highlight important international differences, especially between the situation in the EU and that in the US.

Some of the more important international sources of copyright law which affect the protection of computer programmes are:

Convention for the Protection of Literary and Artistic Works otherwise known as the Berne convention. This dates back to 1886 and establishes such basic principles as recognition of copyright in works authored overseas, and sets out an international minimum term of protection of the life of the author plus 50 years.

[1]Tort is the technical legal term for private law matters such as negligence, trespass and the civil (but not criminal) aspects of assault.

Directive on the Legal Protection of Computer Programs (the Software Directive) Directive 2009/24/EC is the current piece of EU legislation extending copyright to computer programmes. It details the key rights of copyright holders, but also extends particular rights to legal users of software.

Information Society Directive Directive 2001/29/EC harmonises copyright more generally, but has particular provisions which are germane to computer programs. In particular, this directive requires member states to prohibit the circumvention of technological access controls and copy protection of work covered by copyright. In the UK this has been implemented in sections 296-296B of the Copyright Designs and Patents Act 1988. Another important exception to copyright introduced by the Information Society Directive was for temporary copies produced for technical reasons. This placed on a more secure legal footing activities such as web caches and the copying of information which occurs automatically as it moves around computer systems.

1.2.1 The UK and European Law

The UK left the EU on 31 January 2020. Given the important role that EU statute and case law plays in UK copyright law, it is important to address the impact of the UK's exit from the EU on these areas of law. In fact, very little changes in these areas of law in the immediate term.

The departure of the UK from the EU was given domestic legal effect by two UK acts of Parliament, the European Union (Withdrawal) Act 2018, and the European Union (Withdrawal Agreement) Act 2020. Roughly speaking, the first of these statutes gives domestic effect to termination of the UK's previous treaties with the member states of the European Union, while the second gives domestic effect to the Withdawal Agreement[2], a treaty that the UK and the EU concluded dealing with the withdrawal process. The Withdrawal Agreement creates a transition period lasting until at least 31 December 2020, which may be extended by up to two years if the EU and the UK jointly agree this before 1 July 2020. During this transition period, most EU law continues to apply in the UK unchanged. In particular, the impact of EU law on copyright is unchanged during the transition period.

Once the transition period ends, the European Union (Withdrawal) Act provides that most EU legislation applying to the UK continues in force as domestic legislation. This means that none of the legislative rules change unless or until parliament changes them. This means that the legislative basis for software copyright in the UK remains unchanged for the forseeable future. Simiarly, the effect of judgments of the Court of Justice of the European Union on UK law is preserved by the act unless or until they are overriden by statute, or by a subsequent decision of the Court of Appeal or Supreme Court in the UK.

As at the time of writing, the future relationship between the UK and the EU is only in the early stages of negotiations, and it is entirely possible that a future treaty between

[2]Formally, the Agreement on the withdrawal of the United Kingdom of Great Britain and Northern Ireland from the European Union and the European Atomic Energy Community

the parties will have some bearing on the cross-border applicability of copyright law. However such an agreement is unlikely to create a new difference between UK and EU software copyright law.

1.3 Copyright as intellectual property

It might sound strange at first hearing to hear copyright described as a property right. We're used to the concept of property rights, or ownership, in land or in physical objects, but what does it mean to say that a statutory bundle of rights like copyright is a form of property?

To understand this, we need to understand what distinguishes property rights from other legal rights under, for example, contract law or in tort. There are some defining features of property law which we can recognise in copyright:

1. Property rights are enforceable against any third party without that party's consent. This is true for real property: the owner can generally exclude any third party from his or her land, and it's true for copyright: in general all third parties are subject to the exclusive rights of the copyright holder. Conversely, if Alice enters into a contract with Bob, then Charlie is unrestricted by any of the obligations in that contract.
2. Property rights are alienable: they can be transferred in whole or part to other parties. This is true for copyright as well, and the transferee then enjoys the rights on the same basis as the original holder.
3. Property rights can be waived in part by the granting of licences. I waive my right to sue you for trespass by inviting you onto my land. Similarly, the copyright holder may grant licences to copy, distribute, or otherwise deal in copyright works, and a licensee does not breach copyright law by acting in accordance with that licence.

It is important, however, not to take this analogy beyond its scope. A good example of this is theft. Theft is a very particular criminal offence concerned with dishonestly making off with personal property. It is simply not possible to commit this crime with respect to a set of immaterial legal rights. Violating copyright can in some cases be a criminal offence, but it will be an offence under the Copyright Designs and Patents Act 1988 and unrelated to the offence of theft created by the Theft Act 1968.

1.3.1 Other forms of intellectual property

Copyright is not the only form of intellectual property, nor is it the only one which can affect software. However copyright is the form of intellectual property which affects all developers and users of software most pervasively. These are some of the other forms of intellectual property which impact on software but which are beyond the scope of these lectures.

Trademark trademark law restricts the use by third parties of the distinctive names and logos associated with products, including software.

Patent protects the ideas behind inventions. In Europe, software *per se* is not patentable but software which causes physical effects may be patentable as a part of an invention. In the US, a much broader class of software is patentable.

Database rights are *sui generis* rights in the collection of data together to form a database.

Design rights protect the design of objects from copying. These are not directly relevant to software development but can be relevant to graphical designs implemented in software, for example for fonts.

Topography rights are *sui generis* rights in the design of semiconductors. These are of relevance to hardware designers.

1.3.2 Your programme is a book

Copyright applies to far more than software: books, art work, music, and audio and video recordings are among the categories of work which attract the protection of copyright law. The rights associated with copyright vary somewhat according to the class of work so it's important to know to which one your output belongs. Computer programs are quite a lot newer than most copyright law, and so were slotted in later. Computer programs (in both source and object form) are considered to be literary works for the purpose of copyright protection[3]. This is not always a natural fit and has some peculiar effects. Indeed as Boudin J[4] put it in *Lotus Dev. Corp. v Borland Int'l, Inc.*[5]

> Applying copyright law to computer programs is like assembling a jigsaw puzzle whose pieces do not quite fit.

However as we shall see below, recent legislation, particularly the Software Directive, has addressed this somewhat by creating rights and limitations which are peculiar to computer programmes and do not apply to other literary works.

1.4 The rights granted by copyright

The essence of copyright law is the granting to the copyright holder of certain exclusive rights, or monopolies, in the copyright work. These rights belong exclusively to

[3]Software Directive art 1; Copyright, Designs and Patents Act 1988 s 3

[4]In legal writing, the titles of judges are abbreviated and placed after their name. In this case, Boudin J should be read as "Judge Boudin", since Judge Michael Boudin was, at the time, a Judge of the United States Court of Appeals for the First Circuit.

[5]49 F.3d 807, 820 (1st Cir. 1995)

the copyright holder and, subject to only limited exceptions, everyone else requires the copyright holder's permission to do any of these things.

1. The right to make copies whether permanent or temporary, in particular this includes those copies required to load, display and run the program.
2. The right to translate, adapt, arrange or otherwise alter the program. In particular this includes compiling, decompiling and translating into another programming language.
3. The right to distribute to the public, including renting copies of the program.

These rights are laid out in article 4.1 of the Software Directive, and in sections 16-21 of the Copyright, Designs and Patents Act 1988 .

In addition to these primary rights, there are a number of secondary restrictions imposed by copyright law concerned with matters such as dealing in infringing copies. We will not consider these matters here.

1.4.1 Exceptions to copyright

Articles 5 and 6 of the Software Directive lay out several exceptions to the exclusive rights of the copyright holder. Some of these can be really important to the software developer.

The right to use In the absence of specific contractual (i.e. licence) terms to the contrary, someone with a lawfully acquired copy of a piece of software is entitled to use it for its intended purpose and any copies made in the course of this (such as when the program is loaded into RAM) will not infringe copyright. This even extends to modifying the program to correct errors.

The right to make a backup Someone with the right to use a piece of software is also entitled to make a backup copy where that is necessary for the use. This is quite a limited right.

The right to observe, study and test "The person having a right to use a copy of a computer program shall be entitled, without the authorisation of the rightholder, to observe, study or test the functioning of the program in order to determine the ideas and principles which underlie any element of the program if he does so while performing any of the acts of loading, displaying, running, transmitting or storing the program which he is entitled to do."[6] This right would appear to enable the use of debuggers and other monitoring tools.

The right to decompile In limited circumstances, it is legal to copy and translate an otherwise legally held copy of a piece of software in order to achieve interoperability with an independently created piece of software. This right is restricted

[6]Software Directive art 5.3

to those actions needed to achieve interoperability, and the relevant information must not be readily available.

The right to use is essentially a default term in every software licence which may be modified in the licence (for instance to prohibit commercial use of software). The other rights in this section hold for all legal users of software *even if the software licence says otherwise*. This is required by Software Directive art 5 and is implemented by Copyright, Designs and Patents Act 1988 ss 50A-50C & 296A.

1.4.2 The term of copyright

Copyright law is usually described in terms of a trade-off in which authors are encouraged to produce works by the economic benefits they can derive from the copyright in those works, and in return the works will eventually become freely available to all of society. However, the Berne Convention requires a copyright term of the life of the author plus 50 years[7], and EU law now requires that copyright last for the life of the author plus 70 years.[8]. In the case of a work of joint authorship, the duration is measured using the death date of the last surviving author. The effect of these terms in the fast-paced and young field of computer programming is that copyright is, for practical purposes, permanent. To provide some perspective, any surviving computer programmes written by Alan Turing will remain protected by copyright until 2024.

The current long terms of copyright are the result in no small part of extensive lobbying by large commercial copyright owners, typically from fields outside computer programming. The extensive role played by one particular US company in lobbying for the US Copyright Term Extension Act 1998 (also to life plus 70 years) led to opponents terming that law the "Mickey Mouse Protection Act".

1.4.3 Who owns copyright in a work?

Ownership of copyright in a work is governed by Copyright, Designs and Patents Act 1988 s 11, a piece of legislation which is so unusually clear that it is convenient to reproduce the first part of the section here verbatim[9]:

11. First ownership of copyright.
    (1) The author of a work is the first owner of any copyright in it, subject to the following provisions.
    (2) Where a literary, dramatic, musical or artistic work, or a film, is made by an employee in the course of his employment, his employer is the first owner of any copyright in the work subject to any agreement to the contrary.

[7] Berne Convention art 7

[8] Council Directive No. 93/98/EEC Harmonizing the Term of Protection of Copyright and Certain Related Rights 1993 art 1.

[9] The remaining subsection relates to works by public bodies and is not relevant here.

Since students are not employees, copyright in anything a student writes belongs to the student personally. Some universities have copyright policies which claim ownership of their students' work. In the light of the above legislation, it is very unclear on what basis these policies purport to operate.

1.4.4 Copyright applies automatically

It is very common to hear people use the word "copyright" as a verb, and to hear about whether certain works "have been copyrighted". This is based on a fundamental mistake about copyright law which may be rooted in a confusion with patent law (for which registration is a requirement). Copyright exists in a literary work, including a computer program, as soon as it is recorded in writing or otherwise[10]. For computer programs, this means that copyright exists in the source code from the moment it is typed into the computer, and copyright exists in object code the moment it is written out from the compiler.

1.4.5 Penalties and remedies for violation of copyright

Breaching any of the exclusive rights granted under copyright law is a statutory tort which may be enforced through the civil courts[11]. The remedies available include damages or an injunction, for example prohibiting further distribution of infringing copies. There are, in addition, several remedies, such as an order to deliver up the infringing copies, which are particular to copyright infringement[12]. From the perspective of a software startup, the costs of litigation for breach of copyright could themselves be severely problematic, and an order to cease distributing a product which infringes copyright could be ruinous.

In addition to the civil remedies available to copyright holders, deliberate copyright infringement in the course of business or at large scale can be a criminal offence punishable by fines or imprisonment[13].

1.5 The scope of copyright in software

From the perspective of the software developer, there are several important questions about the scope of copyright law and what actions by a developer might infringe copyright. The extent of copyright coverage in software is a matter of interpreting the legislation, a task which falls to the courts. To determine exactly what is and what is not covered, we need therefore to look to the case law.

[10]Copyright, Designs and Patents Act 1988 s 3.
[11]Copyright, Designs and Patents Act 1988 s 96
[12]Copyright, Designs and Patents Act 1988 ss 96-100
[13]Copyright, Designs and Patents Act 1988 s 107

1.5.1 Literal copying

If a developer has access to the source code and copies substantial portions of it without the permission of the copyright holder, then it is clear that he or she infringes copyright. The challenge is in establishing the meaning of the word "substantial", and this has caused the courts some considerable difficulty. One of the leading cases in this regard is *Cantor Fitzgerald v Tradition*[14]. This case concerned two pieces of bond brokering software in the finance industry. The defendant[15] (*Tradition*) had copied portions (2-4%) of the claimant's code. The copied portion consisted of between 2000 and 4000 lines (of VAX BASIC). There were, additionally, claims of non-literal copying which are not relevant here. The test which the judge, Pumfrey J[16] employed was whether the copied code represented a substantial part of the skill and labour of the author. Note that it must be skill *and* labour: a merely mechanical process involving little or no skill will not generate copyright in the work. In this case, there was evidence that the defendants had saved months of skilled work by copying the parts they did. This establishes the principle that copying of a small part of a work may well be "substantial", although it is still unclear how little that might need to be.

1.5.2 Non-literal copying: idea/expression dichotomy

Copyright is often said to protect the expression of an idea, but not the idea itself. For instance article 2 of the Software Directive is:

> Protection in accordance with this Directive shall apply to the expression in any form of a computer program. Ideas and principles which underlie any element of a computer program, including those which underlie its interfaces, are not protected by copyright under this Directive.

This is a slippery distinction which the courts have grappled with over many years. For example is the arrangement of code into a particular configuration of modules and routines part of the expression in a program? In *Cantor Fitzgerald v Tradition*, Pumfrey J rejected this argument for the facts before him as he came to the conclusion that the particular configuration was arbitrary and the result of the same programmers having worked on the two codes rather than being the result of particular skill and labour.

Copying functionality

The courts do seem increasingly firm in the position that functionality is not protected by copyright. In *Navitaire Inc. v EasyJet Airline Co and another*[17] EasyJet had engaged a company called BulletProof to produce a ticketless reservation system which would

[14][2000] RPC 95. As a side note, in the English tradition, the "v" in civil cases is pronounced "and" not "versus".

[15]In civil cases, the party who sues is called the claimant and person being sued is the defendant

[16]In this case, Pumfrey J expands to Mr Justice Pumfrey who was an English High Court judge.

[17][2004] EWHC

be a drop-in replacement for Navitaire's system, which EasyJet was using at the time. In this case, there is no question of literal copying: neither EasyJet nor BulletProof had access to the source code, and the programs were found to be substantially different in their implementation. However they performed the same function and BulletProof's software was designed to have an essentially identical user interface to the original. This case was also heard by Pumfrey J and he had a very clear position on the distinction between functionality and expression:

> Copyright protection for computer software is a given, but I do not feel that the courts should be astute to extend that protection into a region where only the functional effects of a program are in issue. There is a respectable case for saying that copyright is not, in general, concerned with functional effects, and there is some advantage in a bright line rule protecting only the claimant's embodiment of the function in software and not some superset of that software. The case is not truly analogous with the plot of a novel, because the plot is part of the work itself. The user interface is not part of the work itself. One could permute all the letters and other codes in the command names, and it would still work in the same way, and all that would be lost is a modest mnemonic advantage. To approach the problem in this way may at least be consistent with the distinction between idea and expression that finds its way into the Software Directive, but, of course, it draws the line between idea and expression in a particular place which some would say lies too far on the side of expression. I think, however, that such is the independence of the particular form of the actual codes used from the overall functioning of the software that it is legitimate to separate them in this way, and not to afford them separate protection when the underlying software is not even arguably copied.[18]

The user interface about which Pumfrey J writes was a text-based interface using commands, a subject to which we will return below. There was also a GUI interface and Pumfrey J found that, although the code to generate GUI screens is not protected as a literary work, the screens themselves may be protected as art work. It is therefore possible that work-alike programmes which employ very similar graphical interfaces, possibly copying custom icons and the like, will infringe copyright in this way. However the scope for even this sort of infringement is limited. In *Nova Productions Limited v Mazooma Games Limited & Others and Bell Fruit Games Limited*[19], the parties were all manufacturers of arcade games, and all made games based on pool. Nova alleged that various aspects of the respondents' games violated the artistic aspects of its games. These included the way in which the cue was aimed, the appearance and behaviour of a power meter and features such as images of coins rolling across the screen. The Court of Appeal followed the reasoning in *Navitaire* as far as functionality went, ruling that any broad similarities in game play were not protected. The court also had a very restrictive interpretation of what would constitute artistic copying. In particular, the games could not be protected as videos, so any infringement would have to occur in the individual frames considered as art works. In this case, the actual bitmaps used were not all that similar and so no infringement had occurred.

[18]ibid. para 94.

[19][2007] EWCA Civ 219

1.5.3 APIs and languages

As noted above, in *Navitaire* Pumfrey J ruled that textual programming interfaces are not protected by copyright. This question has recently been addressed in high profile cases on both sides of the Atlantic. In the US, Oracle sued Google for producing an unlicensed implementation of Java in the Android operating system. Google won that case at first instance but Oracle had considerable success at appeal[20]. The US supreme court refused to consider an appeal, however the first instance court then ruled in favour of Google on the particular US ground of *fair use*. Oracle have appealed that ruling and the case is still ongoing. In Europe, however, final judgement was handed down by the Grand Chamber of the European Court of Justice in *SAS Institute Inc. v World Programming Ltd*[21]. The core facts of this case are rather similar to the more famous American case. SAS Institute makes statistical business analysis software which is driven by a domain-specific language (the SAS language). World programming built a second independent implementation of this language by examining "learning edition" copies of SAS's software, which they had legally bought. The court came out very clearly in favour of the position that the functional aspects of a computer program, including any inbuilt language and file formats, are not protected by copyright:

> neither the functionality of a computer program nor the programming language and the format of data files used in a computer program in order to exploit certain of its functions constitute a form of expression of that program and, as such, are not protected by copyright in computer programs for the purposes of [the Software Directive].[22]

The reasoning used by the court would lead to the conclusion that a similar argument would apply to APIs and other mechanisms by which a user or another piece of software interacts with the software in question.

There are, however, some important limitations which the court also highlighted. World Programming did not disassemble or decompile the SAS object code. The court noted that, had they done so and used this information to create their implementation, a breach of copyright might have occurred[23]. In this context it is important to note that the decompilation right in the directive[24] is specifically restricted to acts needed to create *interoperability*. World Programming were not creating a piece of software to interoperate with SAS, rather their software was a replacement for it.

Conversely, the court gave a particularly strong meaning to the right to observe, study or test legally acquired software[25]. On the facts, World Programming's execution of their software for study purposes had included actions which fell outside the restrictive "learning edition" licences that they had purchased. The court ruled that these restrictions were overridden by the right to observe, study or test:

[20] *Oracle America Inc. v. Google Inc.*, No. 2013-1021, -1022 (Fed. Cir. 2014)

[21] Case C-406/10 Judgment of the Court (Grand Chamber) of 2 May 2012.

[22] ibid para 46.

[23] ibid para 59-60.

[24] Software Directive art 6.

[25] Software Directive art 5

> the owner of the copyright in a computer program may not prevent, by relying on the licensing agreement, the person who has obtained that licence from determining the ideas and principles which underlie all the elements of that program in the case where that person carries out acts which that licence permits him to perform and the acts of loading and running necessary for the use of the computer program[26]

The court brought together its rulings on the study right and the decompilation right in a particularly clear statement of the law:

> the copyright in a computer program cannot be infringed where, as in the present case, the lawful acquirer of the licence did not have access to the source code of the computer program to which that licence relates, but merely studied, observed and tested that program in order to reproduce its functionality in a second program.[27]

1.5.4 Clean room implementations

The re-implementation of computer software without access to the source is not a new phenomenon. One of the most celebrated US instances of this phenomenon is that of the original IBM PC BIOS. The IBM PC was a machine put together almost completely out of commodity parts. IBM also intended it to be an open platform in the sense that they wanted to encourage third party manufacturers to make add-ons for the PC, so they published full interface specifications for the bus and the BIOS. IBM did not, however, want other companies to be in the market of making IBM compatible computers themselves. IBM came up with an ingenious strategy: they published the full source code for the BIOS in their documentation, but without a licence which would enable a competitor to legally copy the BIOS code in a new machine. By doing this IBM sought to "poison the well" by ensuring that any developer with the expertise to work on a competing BIOS would have seen the IBM source code and could be accused of copying it. IBM did not hold an exclusive licence to the MS-DOS operating system which shipped with the PC, so a competitor who could legally clone the BIOS would be in a position to build compatible computers and could licence MS-DOS from Microsoft to provide a complete system.

Compaq (later followed by Phoenix and others) employed a strategy to defeat this mechanism known as clean room development. They employed some software engineers who were able to demonstrate that they had not had access to the IBM source code. These developers formed the *clean room* and developed the new BIOS code. Meanwhile a separate *dirty room* team had access to the IBM source code, which they used to write program specifications for the clean room team. The dirty room team could then test the code from the clean room and provide feedback until full compatibility was achieved. Although there is no high-level court ruling on this particular case (the legal process ended at a very low level), the fact that a large corporation like

[26] *SAS Institute Inc. v World Programming* para 59.
[27] ibid para 61.

IBM did not elect to litigate this to a high level might be taken as evidence that they did not expect to win. In any event, following *SAS Institute Inc. v World Programming*, clean room implementations in Europe are almost certainly legal.

1.6 Software licences

We have seen above that copyright law grants the copyright holder the exclusive right to authorise the making of copies. In order for copies of software to be legally distributed, whether sold, rented or given away, it is therefore necessary for the copyright holder to grant this permission. This grant of permission is called a *licence*[28]. These licences can be conditional, and indeed usually are. In the closed source world of proprietary software, almost all licences will purport to prevent redistribution and modification. Terms which proscribe certain uses of the software are also common: for instance much software is offered to students and universities at discounted rates on the condition that it is not used for commercial purposes. Such restrictions are usually binding and breaching them is breach of copyright, subject to the specific exceptions found in the Software Directive. Note that in *SAS Institute Inc. v World Programming*, the court did not find that the restrictions in the SAS learning edition licence to be generally invalid, it just found that they did not limit the specific freedoms granted by the directive.

1.6.1 Free and Open Source software

A developer wishing to incorporate proprietary software in his or her work must obtain a licence from the copyright holder to do so and will often need to pay for the privilege. However there is another class of licence in common use which is designed not to extract revenue from copyright, but rather to make the copyright work available to as many people as possible, and to enable those people considerable freedom to do with that software as they please. These licences are known as free software or open source licences, and have their origins in the ideas of an American computer programmer named Richard Stallman. Up until the late 1970s or early 1980s, the proprietary software business model which is now commonplace was far less pervasive, and source code was often freely traded, modified, and passed on again. As this model faded Stallman, then a researcher at MIT, found himself in a position where he was legally unable to participate in the sharing software community he was used to and valued[29]. Stallman had two big ideas, one was to develop an open operating system based on shared code. This became the GNU project which is of huge importance to the history of computing, but is less relevant here. The other was to formulate concept of free software, and consequently of free software

[28]In British English, the verb "to licen**s**e" is spelt with an "s" while the noun "licen**c**e" is spelt with a "c". In American English, both are spelt with an "s". Given that many software licences have American names, this can lead to anomalous, but correct, spelling such as "This software is licen**s**ed under the GNU General Public Licen**s**e, which is our favourite licen**c**e".

[29]For the historical background to the free software movement, see Stallman and Gay (2009) and Levy (2001)

licences. Stallman's concept was articulated as four freedoms which the user of a piece of software must have in order that the software can considered free:

Freedom 0 The freedom to run the program, for any purpose.

Freedom 1 The freedom to study how the program works, and change it so it does your computing as you wish. (Access to the source code is a precondition for this.)

Freedom 2 The freedom to redistribute copies so you can help your neighbor.

Freedom 3 The freedom to distribute copies of your modified versions to others. 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.)[30]

The terminology which Stallman and the Free Software Foundation[31] of which he is president uses is of *free software*. This is an explicitly philosophical description which reflects its proponents view that the usual copyright licensing arrangements are an infringement on the freedom of users and developers to modify and share information. The alternative terminology *open source software* was coined in 1998 by the Open Source Initiative[32] in an attempt to highlight the business case for open development, rather than to advocate openness as a political or philosophical goal. The Open Source Initiative has its own, significantly longer, definition[33] which differs in detail from the FSF's four freedoms, but the practical implications are almost identical: almost all licences which satisfy the FSF definition of free software satisfy the OSI open source definition and vice versa.

1.6.2 If the source is on the web, it's open source, right?

WRONG! This is an incredibly common misconception which can land developers and users of software in enormous trouble. As in the case of the IBM BIOS, merely having made source code available does not in and of itself mean that a third party has a copyright licence to deal in that code. Of course most code which is made available on the web is there because its owner wishes it to be used, but this does not mean that such use is without restrictions. This is a trap which has caught out many developers and distributors of software. An example in point is a library called ParMETIS. ParMETIS is a graph partitioning library which is used by many scientific simulation users who need to divide their computational mesh among the nodes of a distributed memory supercomputer. ParMETIS is produced by a research group at the University of Minnesota and the source is freely available on their website. Many organisations have downloaded the source code and used it for real work, including a prominent UK open source simulation software company. The licence of ParMETIS is:

[30] *Free software definition*, Stallman and Gay (2009) chapter 3.

[31] `http://www.fsf.org`

[32] `http://opensource.org/history/`

[33] `http://opensource.org/osd`. The OSI open source definition is in turn based on the Debian Free Software Guidelines `http://www.debian.org/social_contract#guidelines`.

> ParMETIS is copyrighted by the Regents of the University of Minnesota. It can be freely used for educational and research purposes by non-profit institutions and US government agencies only. Other organizations are allowed to use ParMETIS only for evaluation purposes, and any further uses will require prior approval. The software may not be sold or redistributed without prior approval. One may make copies of the software for their use provided that the copies, are not sold or distributed, are used under the same terms and conditions.[34]

Aside from the awful abuse of the comma, this licence clearly prohibits commercial use (other than for evaluation). The aforementioned UK firm were contacted by the copyright holder's representatives and asked to pay a licensing fee and to kindly forward the details of all of their commercial clients who might also be using ParMETIS. In that case, the company immediately deleted all references to ParMETIS from their software and switched to exclusively using an open source competitor, SCOTCH[35]. The company in question were fortunate that there was an alternative available and that the international nature of the case made enforcement by the copyright holder more expensive and cumbersome than would have been the case domestically.

1.6.3 Permissive and copyleft licences

There are a large number of open source and copyleft licences in current use, and each has its own particular set of conditions which are mostly variations on a few core ideas. Both the FSF and the OSI maintain lists of licences which they consider to be free software and open source licences respectively[36]. However a small number of licences account for a very large proportion of the free and open source software in circulation. Three of these licences, the BSD/MIT licence, the GNU GPL, and the GNU LGPL are both particularly prominent, and illustrate one important difference: that between permissive and copyleft licences.

The BSD licence

The licence of the Berkeley Software Distribution, or BSD licence is a very simple licence which, excluding the warranty disclaimer which follows it, reads:

> Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:
>
> - Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

[34] http://glaros.dtc.umn.edu/gkhome/metis/parmetis/download
[35] http://www.labri.fr/perso/pelegrin/scotch/
[36] http://www.gnu.org/licenses/license-list.html, http://opensource.org/licenses

- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.[37]

In other words, the BSD licence enables anyone to do essentially anything they like with the software other than strip off the copyright notices. This is very much like the academic practice of using others work but giving acknowledgement by citing. In particular, free software released under the BSD licence can be incorporated with software released under any other licence, even a proprietary licence which grants the licensee no rights beyond the right to run the software.

There are several variants on the BSD licence which all have essentially the same legal effect. Some variants of the licence include a clause explicitly prohibiting claims that the original author endorses any modified version of the work, while others are worded differently in various ways. There is also a very old version of this licence which required the original copyright holder to be acknowledged in all promotional materials related to software using the BSD licensed software. This so-called "obnoxious BSD advertising clause" was officially rescinded by the licence's authors, the University of California in 1999. A popular reworded version of the licence is known as the MIT licence.[38]

The GPL

Richard Stallman's GNU project was based on a vision of building a software ecosystem based on sharing code. While the BSD licence is a free software licence in Stallman's terms, it contains no requirements for software sharing. Instead, the Free Software Foundation came up with a creative use of licensing terms which enables copyright law to be used to encourage the sharing of source code. This is often described as a hack on the copyright system, which is more conventionally used to prevent copying of work. The name the FSF uses for licences of this sort is *copyleft*, a play on words which alludes to their unusual use of copyright law.

The flagship copyleft licence, and the one which applies to a huge range of software including the Linux kernel and GCC is the GNU General Public License, or GPL. There are two main versions of this licence in current use, versions 2, issued in 1991, and 3, issued in 2007[39]. Version 3 is a comprehensive redraft of the licence and is much more clearly written. It also contains some additional terms to do with patents, trademark and digital rights management which are beyond the scope of these lectures, but the essential copyright terms of the two licence versions have similar effects.

In summary, the GPL permits those obtaining the code to distribute modified or unmodified versions of the work in source and/or object form provided:

1. that the *whole* of the modified work is itself distributed under the GPL; and

[37] http://opensource.org/licenses/BSD-2-Clause

[38] http://opensource.org/licenses/MIT

[39] http://www.gnu.org/licenses/gpl.html, http://www.gnu.org/licenses/old-licenses/gpl-2.0.html

2. that anyone who is given an object code version of the software has the source code made available to them.

Together these clauses are what make the licence *copyleft*. The effect of these is to create a space of GPL (and GPL-compatible, see below) software in which participants can share code, as long as they make the source available and do so under the GPL. By ensuring that GPL terms must apply to the redistributed work, the licence ensures that downstream users enjoy the same rights as those upstream.

There are some common misconceptions about the GPL, one of the most pervasive of which is that redistributing GPL software obliges one to make the source of the redistributed version publicly available. In fact, the GPL generally only requires that you provide the source code to those to whom you provide binaries[40]. However since all the recipients of your work are then entitled to give out the source code as they see fit, publishing the source code on the Internet it is usually the simplest and easiest way to comply with the GPL.

The LGPL

The GPL by its nature restricts the ability of developers to combine GPL and proprietary code. This was a deliberate policy choice by the FSF: it creates an environment in which those who wish to benefit from using GPL software in software they distribute must contribute that software to the GPL ecosystem. Many developers like this arrangement and choose to write GPL software. However many other developers wish to, or feel for market reasons that they need to, allow their downstream users to combine their software with software which is not GPL and distribute the result, for example by linking against proprietary libraries and shipping a binary which is a mixture of free and non-free software.

To cater for this use case, the FSF produced the GNU Library General Public License, which they renamed in later versions the GNU Lesser General Public License (in both cases abbreviated as LGPL). Versions 2.1 of 1999 and 3 of 2007 are commonly in use[41]. The LGPL employs a similar set of rules about redistribution of modified code to the GPL, indeed LGPL version 3 explicitly incorporates all of the terms of GPL 3. In addition, the LGPL permits the distribution in binary form of works produced by combining LGPL and non-LGPL code by compiling and linking them. In order to distribute such a program under the LGPL, the source of the LGPL part must be provided and a mechanism for relinking with a (possibly modified) recompiled version the the LGPL part must be provided.

[40]For full details see GPL V3 clause 6

[41]`http://www.gnu.org/licenses/lgpl.html`, `http://www.gnu.org/licenses/old-licenses/lgpl-2.1.html`

1.6.4 Licence compatibility

Suppose a developer wishes to incorporate code from two or more different sources, available only under different licences in his or her work, and distribute the results. Is this legal, and if so, which licences can the developer use? The answer is that the developer must obey the licences on all the code he or she uses, and so can redistribute the work only under a licence which imposes all the restrictions required by each of those other licences.

For the licences we considered above, there is a neat inclusion relationship which is summarised by table 1.1

	May be distributed under		
Code under	BSD	LGPL	GPL
BSD	Yes	Yes	Yes
LGPL	No	Yes	Yes
GPL	No	No	Yes

Table 1.1: Compatibility matrix for the BSD, GPL and LGPL licences

For other licences, the answer will depend on the terms of the licences involved. The FSF maintains a listing of the licences they consider to be compatible with the GPL[42] (by which they mean that code under those licences may be redistributed under the GPL) but for other combinations it is a question of reading the licence text. A graphical version of part of the FSF compatibility matrix is shown in figure 1.1. Notice that GPL 3 is not compatible with LGPL2. However, many authors add a so-called "any later versions" clause to their GPL copyright notice which allows redistributors to switch to a later GPL version. The Free Software Foundation's suggested wording for a GPL copyright notice has this clause:

> <one line to give the program's name and a brief idea of what it does.>
> Copyright (C) <year> <name of author>
>
> This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.
>
> . . . [warranty exclusion omitted].[43]

1.6.5 Linking and copyright

What is the effect in copyright law of linking against a library? Is the resulting program a single work which contains both the program and the library, or are the program and library separate works in object form even when combined together?

[42] http://www.gnu.org/licenses/license-list.html
[43] http://www.gnu.org/licenses/gpl.html

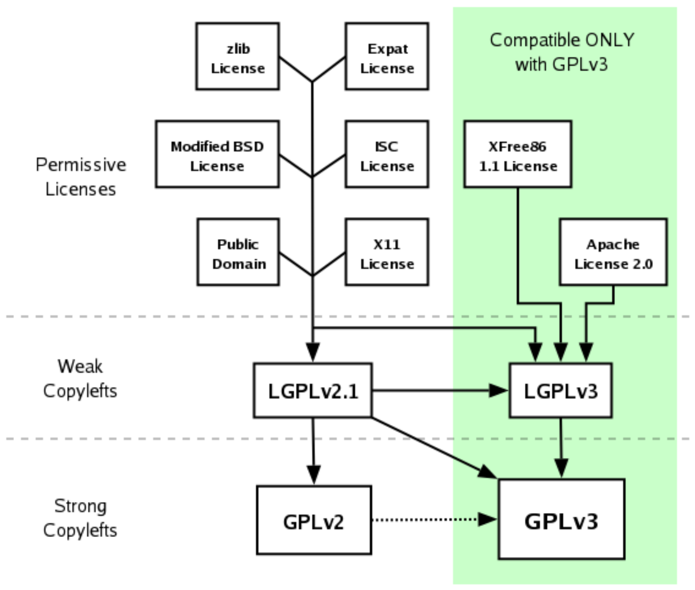

Figure 1.1: GPL compatibility matrix. Note that GPL 3 is GPL 2-incompatible. Figure from `http://www.gnu.org/licenses/quick-guide-gplv3.html`.

Perhaps surprisingly, there does not seem to be any case law on this question, however many legal authors have considered this question, especially in the context of the GPL.

The question of linking, and dynamic linking[44] in particular, arises in the context of the GPL because the FSF asserts that if a GPL library is linked against a program, then binary redistribution of that program is only possible under the GPL[45]. This creates issues for, for example, the authors of proprietary binary-only Linux kernel modules, such as the NVIDIA graphics driver.

The majority position among legal writers appears to be that dynamically linking against a library does not infringe the copyright in that library, and therefore the licence of the library is not relevant to the licence on the main program[46]. Here I will just briefly survey the arguments for this. As a concrete example, have in mind the classic C situation in which there is a library compiled as a shared object (Linux `.so`, Windows `.dll`), accompanying headers which define data structures and function prototypes, and a main program which is to be compiled using those header files and linked dynamically against the library at run time.

Is the source code of the main program an adaption or copy of a substantial part of the library? The only part of the library which is reproduced in the source code is the names of the functions and external variables which are used in the main program. Under *SAS Institute Inc. v World Programming*, these are functional, interface compo-

[44]Dynamic linking in this context might be understood as extending to other run-time linking operations such as those that occur in Java and in interpreted languages such as Python.

[45]See, for example, `http://www.gnu.org/licenses/gpl-faq.html#LinkingWithGPL`

[46]See, for example, van Holst (2013), Katz (2007), Lindberg (2008), and Rosen (2004). The latter works are American but the arguments presented do not appear to rely on any aspect of copyright law which differs between the UK and the US.

nents which are not protected by copyright. It is therefore difficult to see a basis on which the main program source code could be found to infringe the copyright of the library (and therefore be affected by the licence terms on the library).

Now consider the compiled binary main program, on disk and therefore not currently linked against the library. This contains a translation of the source code and the header files. As noted above, the source code is not affected by the copyright in the library but the header files are directly taken from the library. It now matters what is in those header files: suppose they are typical C header files and only contain interface information such as function prototypes and simple type declarations. The ruling in *SAS Institute Inc. v World Programming* would once again appear to indicate that there is no copyright protection for these sorts of header files. On this basis, the binary contains no copyright protected material from the library, and so distribution of this binary would not infringe copyright. The final point in the chain is the execution of the program, at which point the dynamic linker matches the memory copy of the executable with the memory copy of the library. *If* this is considered by a court to create a new work in the computer's memory, then this might potentially infringe copyright (by making an adaptation of the library) but the Software Directive[47] makes it clear that copies made in order to run the program do not infringe copyright in the absence of licence terms to the contrary, and the GPL explicitly does not place any restrictions on the running of software[48].

1.6.6 Code in headers and static linking

The arguments above can be generalised to a wide variety of circumstances: including for interpreted and byte-compiled languages. However there are some circumstances which are a little different from the dynamic linking situation. First, if the program is statically linked rather than dynamically linked, then the compiled code of the library is included in the compiled binary program, and so copying the binary outside the terms of the library licence will infringe copyright. In this circumstance it seems likely that the GPL licence on the library does make distribution of the binary program only possible if the program is released under GPL-compatible terms.

Another similar situation occurs where the header files contain executable code as well as interface information. If this executable code has sufficient expressive quality that it is covered by copyright then use of the headers is similar to the static compilation situation: the compiled binary contains the compiled header code and therefore must comply with its licence.

1.7 Further reading

A leading UK text on digital copyright generally, including much more than copyright in software, is Stokes (2019). A comprehensive, although US law-based, guide to

[47]Art 5.1
[48]GPL v3 clause 2.

working with copyright and patent in an open source context is provided in Lindberg (2008). The particular case of linking and the GPL is discussed in depth in van Holst (2013). Informative discussion of the development of copyright law in the digital context is to be found in Lessig (2006).

Lecture 2

An introduction to contract law

2.1 What is contract?
. . . and why would a computer scientist care?

A contract is a legally enforceable agreement. The making and enforcing of contracts is a core feature of society: it would be impossible for a sophisticated economy to function without the ability to enforce at least some agreements. We each create several contracts every day, most obviously when we buy items. Those of us who are employed or who work as contractors depend on contracts in order to be paid.

Computer scientists encounter contract law in two ways. First, as participants in the economy like all others. This is particularly relevant for the many computer scientists who create start-up companies to market their ideas. However computing also creates special legal situations: where is the contract in a web service? What's the legal effect of a "click through" licence. In this lecture, we will just dip our toes into this area to attempt to gain a feeling for the issues involved.

2.2 A brief diversion into sources of law

Unlike copyright, the law of contract is not, by and large, standardised by international treaties. Contract law varies from jurisdiction to jurisdiction and, although the basic character of a legally enforceable agreement has considerable similarities between jurisdictions, the details vary very considerably. Since Imperial College is in London, we will primarily consider our home jurisdiction, which is England and Wales. Even though Scotland is also part of the United Kingdom, it has its own legal system with a rather different law of contract. Conversely, England and Wales is a more complete part of the common law family, so there is substantial commonality between its contract law and that of other common law jurisdictions, such as Ireland, many Commonwealth countries and the US.

2.2.1 Civil codes

In many countries, particularly those influenced the legal traditions of major continental European countries, the core source of private law rights and obligations such as contract is a civil code. That is, a law passed by parliament which lays out the rules of civil law. The most famous examples of this approach include the *Bürgerliches Gesetzbuch* in Germany and the *Code Napoléon* (officially *Code Civil*) in France. This approach has been very widely adopted in those parts of the world which were not colonies of Britain. For example Japan's modern civil law was rewritten at the end of the nineteenth century based on the Bürgerliches Gesetzbuch while, unsurprisingly, many former French colonies retain civil codes on the French model.

2.2.2 Precedent and the common law

The situation is very different in England and Wales, and in most of the countries which have at some point been colonies of Britain. Common law countries have no document taking the role of a civil code: there is simply no equivalent. No law passed by parliament establishes the binding force of contracts at law. Where, then, does contract law come from? The answer is that the basis of the common law is the decisions of previous courts. In this manner, the law of contract, negligence, property and even the criminal law have gradually emerged over countless cases stretching back to the middle ages.

The consistency of this position rests chiefly on the rules of precedent: courts should decide cases in a manner consistent with previous decisions. Within a single court hierarchy, courts *must* follow the previous decisions of higher courts in the hierarchy. For this reason, higher courts allow appeals from lower courts in order to provide a definitive solution in areas in which the law is unclear.

2.2.3 Statute

Of course the absence of a civil code does not mean that there are no statute laws passed by parliament. Parliament can and does pass laws on many matters, and when it does so these override the common law. For example larceny (theft) was a common law offence in England and Wales until 1916 when the Larceny Act 1916 codified the law of theft as a series of statutory offences. The law of copyright, which we covered in a previous lecture, is likewise primarily statutory. In contrast, contract is primarily common law, although it does intersect with areas such as competition law in which statute plays a greater role.

Just as parliament may intervene to change the common law by statute, the courts have an important role in interpreting and applying statute law. In doing this, the same rules of precedent apply, so there develops a common law of how a particular statute is to be interpreted.

2.3 Elements of contract

A contract is a legally enforceable agreement. While it is clearly a good and necessary thing that some agreements are enforced, the law must decide which agreements these are to be, and what constitutes an agreement at all.

2.3.1 Agreement

The most basic component of contract is agreement. The parties must agree on the terms of the contract. The usual analysis of agreement is by offer and acceptance. One party makes an offer "I will sell you my old laptop for £200" and the other indicates his agreement. It is important to note that this agreement creates the contract, which is then binding from that point. An offer in circumstances where it is clear that a purported acceptance might not be itself accepted is not an offer in these terms. This is particularly important in retail sales contracts, whether in shops or online. Where goods are displayed in a shop, or on an e-commerce website, they might be described as offered for sale. However the courts have come to the conclusion that this is not the case. In *Pharmaceutical Society of Great Britain v Boots Cash Chemists (Southern) Ltd*[1] the (then new) concept of a self service chemist raised the problem of whether exposing the goods on the shelves constituted an offer to sell. If it did, then the customer picking up the item and placing it in his basket would constitute acceptance and create a binding contract which would merely be executed by the exchange of money for property at the checkout. In this case the issue was that a sale so formed would not have been supervised by a pharmacist, in violation of relevant legislation. The court instead ruled that displaying the goods was merely an *invitation to treat*. In other words the customer was invited to make an offer to buy the goods at the marked price, which the shop might or might not accept. The contract was therefore completed at the time at which the cashier accepted the offer to buy. A similar conclusion was reached in *Fisher v Bell*[2] in which the court ruled that displaying a flick-knife in a shop was not offering it for sale (which would have been illegal), and in *Mella v Monahan*[3] in which the same result was reached for obscene pictures. The much older case of *Grainger & Son v Gough (Surveyor of Taxes)*[4] established that catalogues are also offers to treat, and not offers to sell. For e-commerce, this well established field of law creates the situation that it is the buyer clicking on the relevant order button which constitutes the offer, and the contract is formed by the vendor accepting that offer.

2.3.2 Communication

To achieve the level of agreement, sometimes termed *meeting of minds* that contract requires, it is necessary that the offer is communicated to the offeree, and that acceptance is communicated to the offeror. With the exception of a rule to do with

[1][1953]EWCA Civ 6
[2][1961] 1 QB 394
[3][1961] Crim LR 175
[4][1896] AC 325

the physical post, which is less relevant here, the communication must be actually effective so that the offeree actually receives the offer and the offeror actually receives the acceptance. The situation with respect to electronic communication was rather helpfully stated by Denning LJ in *Entores Ltd v Miles Far East Corporation*[5]

> In all the instances I have taken so far, the man who sends the message of acceptance knows that it has not been received or he has reason to know it. So he must repeat it. But, suppose that he does not know that his message did not get home. He thinks it has. This may happen if the listener on the telephone does not catch the words of acceptance, but nevertheless does not trouble to ask for them to be repeated: or the ink on the teleprinter fails at the receiving end, but the clerk does not ask for the message to be repeated: so that the man who sends an acceptance reasonably believes that his message has been received. The offeror in such circumstances is clearly bound, because he will be estopped from saying that he did not receive the message of acceptance. It is his own fault that he did not get it. But if there should be a case where the offeror without any fault on his part does not receive the message of acceptance - yet the sender of it reasonably believes it has got home when it has not - then I think there is no contract.
>
> My conclusion is, that the rule about instantaneous communications between the parties is different from the rule about the post. The contract is only complete when the acceptance is received by the offeror: and the contract is made at the place where the acceptance is received.

There are a number of statements here which are relevant in e-commerce settings. The key point is the last one, that the communication must actually occur. The exception noted is one in which the acceptance is not communicated due to fault on the part of the offeror. This refers to a part of the law called *equity* which we will not consider further: suffice it to say that both equity and consumer protection law are likely to impose on the negligent vendor who manages to inflict damage on his or her customers. It should also be noted that the statement about the location where the contract occurs (which has implications for which law governs the contract) may be affected in the consumer case by European and domestic consumer protection legislation.

Unilateral contracts

In some circumstances the offeree need not communicate his acceptance to the offeror, instead her conduct by actually performing her obligations under the contract suffice. The leading case in this area is *Carlill v Carbolic Smoke Ball Company*[6]. In this case, the Carbolic Smoke Ball Company had placed a newspaper advertisement stating:

> £100 reward will be paid by the Carbolic Smoke Ball Company to any person who contracts the influenza after having used the ball three times

[5][1955] EWCA Civ 3

[6][1892] EWCA Civ 1

> daily for two weeks according to the printed directions supplied with each ball…£1000 is deposited with the Alliance Bank, shewing our sincerity in the matter.

Mrs Carlill used the ball as directed but still got the flu. Among the many arguments which the Carbolic Smoke Ball Company had raised were that the plaintiff had not communicated her acceptance of the contract to the company. It was held that in contracts of this type, this is not a requirement.

This form of contract is significant in computing not just because it provides a legal basis for initiatives like XPRIZE[7] but because it applies analogously to situations such as vending machines where a contract may be made by the offeree interacting with an automated and autonomous system out of communication with the person making the offer.

2.3.3 Consideration

An aspect of contract law which is unique to the common law jurisdictions is the doctrine of "consideration". This does not have anything to do with "consideration" in the sense of thinking about something. Instead, the word is used in the sense of "in consideration of your action A, I will do B". The doctrine of consideration states that a valid contract must be an exchange of promises in which both parties agree to do something. A purely gratuitous act, in other words a pure gift, is not a contract[8]

The traditional authority for the doctrine of consideration is the judgement of Lord Dunedin in *Dunlop Pneumatic Tyre Company, Ltd v Selfridge & Co., Ltd.*[9]:

> My Lords, I am content to adopt from a work of Sir Frederick Pollock, to which I have often been under obligation, the following words as to consideration: "An act or forbearance of one party, or the promise thereof, is the price for which the promise of the other is bought, and the promise thus given for value is enforceable." (Pollock on Contracts, 8th ed., p. 175.)

The consideration need not be monetary (indeed only in contracts to lend money is it usual for *both* parties to be providing money as consideration), and it need not actually pass from one party to the other. For example in *Hamer v. Sidway*[10] an William E. Story promised his nephew (confusingly also named William E. Story) that would pay him $5000 if "he would refrain from drinking liquor, using tobacco, swearing, and playing cards or billiards for money until he should become 21 years of age". The nephew did so, and successfully sued for the money. The court ruled that refraining from doing things which he was perfectly legally entitled to do constituted sufficient consideration on the part of the nephew.

[7]`http://www.xprize.org/prize-development`

[8]A pure gift can be made legally effective using a more formal legal process called a deed, however this is beyond this lecture.

[9][1915] UKHL 1, [1915] AC 847

[10]124 N.Y. 538, 27 N.E. 256 (N.Y. 1891)

It is also important to note that consideration need not represent the full market value of the consideration promised by the counter party. For example, a common legal trick to transform a one-way gift into a legally binding contract is for the other party to promise to pay one peppercorn. For example in Chappell & Co Ltd v Nestle Co Ltd[11], Nestle were running a promotion in which members of the public could send three chocolate bar wrappers and 1s 6d (7.5p) to receive a record. The court ruled that the chocolate bar wrappers did constitute part of the consideration of the contract, despite having trivial inherent value and being of no actual use to Nestle.

2.3.4 Binding consideration and options

The promise of consideration must actually be binding on the promisor. Conditional promises of the form "if you do A then I will do B" are simply an offer, which the counter party can accept and be bound to by doing A, thereby creating a unilateral contract. This distinction is important, because a mere offer can be withdrawn at any time up until acceptance, while a contract is immediately binding. The mechanism for making binding contracts out of conditional offers is the *option*. An option contract has the form "I will pay you £x in return for the promise that if I do A then you will do B". Now the first party has provided consideration in the payment of £x while the second party provides consideration by being bound to do B if the first party does A. Note that in this circumstance, neither party has an obligation which is purely optional for themselves, both are actually bound to do something. Of course the parties can agree any valid consideration for an options contract, it need not be money.

2.3.5 Other elements of contract

There are other elements of contract, and reasons why contracts or parts of them may be invalid, which go beyond the scope of this short lecture. Contracts may be void if there terms are too uncertain or if they are illegal (for example in breach of competition law). One party may be able to escape a contract if they lack the capacity to make contracts (for example in the case of children) or if the contract was based on misrepresentations. Some less formal agreements, especially in social or family settings, may not be contracts because the parties do not intend to be legally bound by their agreement. For a more full treatment, readers are referred to, for example, C. Monaghan and N. Monaghan (2013).

[11][1960] AC 87

2.4 Remedies for breach

2.4.1 Damages

The usual remedy for breach of contract, and the only one to which a claimant has an unconditional right upon proving breach of contract, is damages. This means that the defendant has to pay a sum of money to the claimant sufficient to place the claimant in the same position he or she would have been in had the contract been correctly fulfilled.

2.4.2 Specific performance

Specific performance is a remedy from the law of equity constituting an order that the breaching party must carry out its obligations under the contract. Specific performance is awarded at the discretion of the court and only in cases in which damages would not provide an adequate remedy: for example because alternative sources of the promised goods or services are not available.

2.4.3 Injunction

Like specific performance, an injunction is an equitable remedy which the court may award if damages are not adequate to the case. An injunction is an order to do or abstain from doing something. The scope of injunctions can be very wide indeed.

2.4.4 Liquidated damages and the rule against penalties

Ascertaining what the actual damages are may itself be a complex procedure. The parties may therefore decide to include in the agreement a clause specifying the amounts to be paid in a particular case. For example, if delivery is late, the contract may provide that a particular sum is to be paid. This is permitted to the extent that the amount to be paid represents an estimate of the damages which would be suffered were that breach to occur. These sums of money are termed "liquidated damages". However if the amount is instead intended to penalise the breaching party and thereby to induce him or her to comply with the contract, this is not permitted. The law does not permit parties to establish what would in effect be private systems of justice backed by fines.

The leading case in this area is *Dunlop Pneumatic Tyre Co Ltd v New Garage & Motor Co Ltd*[12]. Dunlop had a blatantly anticompetitive (although not at the time illegal) minimum price clause in its contract to supply tyres to retailers. The contract specified a £5 per tyre liquidated damages payment if tyres were sold below the specified price. The significant point here is that the actual damage to Dunlop was very hard

[12][1914] `http://www.bailii.org/uk/cases/UKHL/1914/1.html`UKHL 1, [1915] AC 79

to quantify, as it was indirect. The court found that the liquidated damages could be enforceable even in the face of uncertainty and that “provided that figure is not extravagant there would seem no reason to suspect that it is not truly a bargain to assess damages, but rather a penalty to be held *in terrorem*.”

Bibliography

van Holst, W. H. (2013). “Less may be more: copyleft, -right and the case law on APIs on both sides of the Atlantic”. In: *International Free and Open Source Software Law Review* 5.1, pp. 5–13.

Katz, A. (2007). “GPL – the linking debate”. In: *Magazine of the Society for Computers and Law* 18.3, pp. 13–16.

Lessig, L. (2006). *Code: Version 2.0*. Lawrence Lessig.

Levy, S. (2001). *Hackers: Heroes of the computer revolution*. Vol. 4. Penguin Books New York.

Lindberg, V. (2008). *Intellectual Property and Open Source: A Practical Guide to Protecting Code*. O’Reilly.

Monaghan, C. and N. Monaghan (2013). *Beginning Contract Law*. Routledge.

Rosen, L. (2004). *Open source licensing*. Prentice Hall PTR.

Stallman, R. M. and J. Gay (2009). *Free software, free society: Selected essays of Richard M. Stallman*. CreateSpace.

Stokes, S. (2019). *Digital copyright: law and practice*. Hart. URL: http://dx.doi.org/10.5040/9781509917327.