

# Scope of copyleft clause under Polish law

Krzysztof Siewicz

## Table of Contents

1. Introduction.....	1
2. Copyleft – all rights reversed.....	2
3. Scope of copyleft under GNU General Public License.....	3
4. Code reuse basics.....	5
5. Scope of copyleft under American law.....	6
6. Determinants of copyleft under Polish law.....	10
A. General remarks.....	10
B. Idea-expression dichotomy.....	11
C. Fair use.....	13
D. Derivative works.....	14
E. Collections, collective works, joint works.....	15
7. Conclusion.....	16

## 1. Introduction

In 1984, a programmer, ideologist and a man of vision, Richard M. Stallman set off for his moral crusade for the rights of computer users. These rights are four freedoms, which form the Free Software Definition formulated by him:

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

The freedom to study how the program works, and adapt it to your needs (freedom 1) ...

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

The freedom to improve the program, and release your improvements to the public, so that the whole community benefits (freedom 3) ... .<sup>1</sup>

It follows from the definition that „freedom” of software is first and foremost its public availability and the prohibition against allowing to control it by any individual. Stallman's aim is to create a complete free software system, named „GNU”. Initially, Stallman worked on the system alone, but subsequently more and more programmers have started to release their programs according to the rules set forth in the Free Software Definition. Currently, mostly after the release of Linux kernel by Linus Torvalds in 1991, the GNU system is being developed by thousands of programmers around the World.

Stallman is aware that the predominant practice of intellectual property protection reaches for the end opposite to the one set forth by the freedoms advocated by him. Major commercial players in the ICT market forcefully defend „their” copyrights, and have recently started to eagerly embrace the more efficient patent protection. Where Stallman would like to see a commons field, strict lines of influence are being drawn, resembling to some the historical enclosure movement. Thus, Stallman designed a model of legal protection named „copyleft”, aimed at guaranteeing the

---

<sup>1</sup> Richard M. Stallman, *Free Software Definition*, 41 in: RICHARD M. STALLMAN, *FREE SOFTWARE, FREE SOCIETY: SELECTED ESSAYS OF RICHARD M. STALLMAN* 41 (GNU Press, Boston, 2002) All publications of Stallman are also available at: <http://www.fsf.org>.

freedom of software meant to be free.

## 2. Copyleft – all rights reversed

The necessity for designing special legal schemes for the protection of free software may be explained by the following reasons. First and foremost, Stallman's freedoms are just a proposal, while the current legal system grants authors the monopoly in all activities covered by the freedoms. Without the authorization of the rights holder, it is impossible to copy, modify or distribute a computer program.

Moreover, a simple disposal of rights, resulting in placing it into the public domain, would not secure its freedom for too long. Anyone who modified such a program would become the copyright holder in the new work created in such way; thus, the program would be again appropriated. This perhaps requires a longer explanation. Certainly it is impossible to appropriate a work that has fallen into the public domain; however, what can be appropriated is the new version of such program created by its modification or use in a new program. The rights to such new program would vest in their author. Since in the software market there is a significant demand for new, improved programs, the old version being in the public domain would be of low value.<sup>2</sup> The market would be taken by the new program, and its creator would benefit from the work of the persons who contributed the original to the public domain.

In order to overcome such obstacles in the development of free software, Stallman proposed „copyleft”, which is „the copyright flipped over to serve the opposite purpose”, since instead providing means to privatize a work, it forces to respect its freedoms.<sup>3</sup> Generally speaking, „copyleft” boils down to granting to all the rights resulting from the four software freedoms together with the prohibition to impose any restrictions of these freedoms. According to Stallman, „copyleft” makes these freedoms inalienable.<sup>4</sup>

Legally speaking, „copyleft” is a part of software license. The best known and most popular free software license is GNU General Public License, version 2.0 (hereafter „GPL”), which grants users all four software freedoms. Its „copyleft” clause reads:

You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.<sup>5</sup>

The rights to copy, modify, prepare derivative works of and distribute the program granted by the GPL are conditioned upon the observance of the „copyleft” clause above. The licensee cannot distribute programs covered by the scope of „copyleft” under the conditions different than set forth

---

<sup>2</sup> The proof is the popularity of Apache server, based on the public domain program httpd (<http://www.apache.org>).

<sup>3</sup> Richard M. Stallman, *The GNU Project*, 20 in: RICHARD M. STALLMAN, *FREE SOFTWARE, FREE SOCIETY: SELECTED ESSAYS OF RICHARD M. STALLMAN*, 15 (GNU Press, Boston, 2002).

<sup>4</sup> *Id.*

<sup>5</sup> GPL Sec. 2 b), <http://www.fsf.org/licenses/gpl.html>.

in the GPL. Such effect is commonly referred to as the „copyleft virus”, since it leads to infecting with freedom all programs that are in a certain relation to the original free program. This is how the „flipping the copyright over” works. It is the intent of the „copyleft” drafters to prevent the licensees from using their copyright monopoly in order to limit Stallman's four freedoms.

The article attempts to estimate what is the „copyleft” clause under Polish law. It is limited to the analysis of the GPL's „copyleft” since it has become a model license contract, covering roughly 70% of free software currently available;<sup>6</sup> however, there are many other standard contract forms available that include similar „copyleft” clauses.<sup>7</sup>

In order to estimate the exact scope of „copyleft” it is necessary to carefully analyze the GPL itself. Software writing techniques definitely have important influence in the construction of this clause too. Additionally it has to be kept in mind that the GPL was drafted under American law and directly references to its institutes. Moreover, guidelines for the interpretation should be looked for in the software trade practice and customs of the hacker community.<sup>8</sup> All of these elements will be analyzed before the attempt to analyze „copyleft” under Polish law is made.

### 3. Scope of copyleft under GNU General Public License

It follows from GPL Sec. 2 b) that „copyleft” covers works in whole or in part containing or derived from the program or any part thereof. This is explained in more detail in the remainder of Sec. 2:

[Sec. 2 b)] requirements apply to the modified work as a whole. If identifiable sections of that 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.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, **the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.**

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program)

---

<sup>6</sup> Data according to SourceForge, <http://www.sourceforge.net>.

<sup>7</sup> Free Software Foundation publishes a long list of licenses, sorted using the criteria of „freedom”, the use of „copyleft” clause and compatibility with the GPL (<http://www.fsf.org/licenses/license-list.html>). The compatibility with the GPL is defined as the legal possibility to link a program under a compatible license with a program under the GPL.

<sup>8</sup> In this article, the term „hacker” will be used to describe a highly qualified programmer. The reader who wished to know more about the hacker culture should read on-line papers by Stallman, Eric S. Raymond and other activists of free software movement (Free Software, Open Source Software). Given the scope of this article, the analysis of customs and practice will be limited to the rules contained in „GPL Frequently Asked Questions” published by the Free Software Foundation. (<http://www.fsf.org/licenses/gpl-faq.html>).

on a volume of a storage or distribution medium does not bring the other work under the scope of this License.<sup>9</sup>

Additionally, „a work based on the Program” is treated by the GPL to be the modification of the program's copy or of its part („You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program”).<sup>10</sup>

A general observation that follows, is that the basic delimitation of the scope of „copyleft” is the author's right to control derivative works as well as the right to the works used in collective works.<sup>11</sup> However, it is not easy to determine what they exactly are if applied to computer programs. Software enters into various relations with hardware, the user and other software. It is common to write software using programs already written (so called „code reuse”), and these techniques constantly develop. Therefore, „copyleft” clause does not only refer to copyright law provisions. On the other hand, it would be not possible to set its scope in too much detail, since it would force to amend the GPL after every development in informatics.

Nevertheless, the part of GPL quoted above attempts to specify the scope of „copyleft” clause further, in addition to referring to copyright law institutes. Vetter points out two expressly indicated states not covered by the clause: „mere aggregation” and „identifiable separate and independent sections”, and reads from the license a three-step test to determine which fragments are exempted: (1) fragment is not based on the free program; (2) it can be reasonably considered a separate and independent work; and (3) it is distributed as a separate work.<sup>12</sup> As it shows, the specification of the GPL is generally in-line with the understanding of „collective works” and „derivative works”.

The same author, after analyzing the GPL enumerates four categories of relations between programs and excludes from the scope of „copyleft” only the fourth.<sup>13</sup> These categories are: (1) software in a single container or file modified or extended by adding code from other source; (2) software in different containers designed to work under the control of one process, linked at production or at the time of execution; (3) software working under different processes, independent but in certain relations; and (4) separate software on a single distribution medium.<sup>14</sup> Regardless of following the classification above, it is necessary to describe the variety of possible factual states effecting from the use of various software writing techniques that subsequently have to be faced with the law.

---

<sup>9</sup> GPL Sec. 2, <http://www.fsf.org/licenses/gpl.html> (emphasis added).

<sup>10</sup> *Id.*

<sup>11</sup> *Cf.* Greg R. Vetter, „*Infectious*” *Open Source Software: Spreading Incentives or Promoting Resistance?*, 36 *RUTGERS LAW JOURNAL* 53, 69 (2004) (pointing to the right to copy and distribute as the alternative sources of obligations from the „copyleft” clause).

<sup>12</sup> *Id.* 95-96.

<sup>13</sup> *Id.* 124. It should be repeated after Vetter (*id.*) that in case of the Linux kernel other categories may also be sometimes excluded from the scope of „copyleft” due to the reservation made by its author described below, which may be considered the modification of original GPL terms (see text accompanying FN. 34).

<sup>14</sup> *Id.* 109.

#### 4. Code reuse basics

Good programmers know what to write. Great ones know what to rewrite (and reuse).<sup>15</sup>

The author does not wish to discuss whether software development is an art or a craftsmanship. It is true; however, that apart from writing their own code, much of the programmers' effort goes into deciding which already developed components to use in a new program, and how to do it in order to use the computer resources in the most efficient way.

The easiest way is to modify an existing program realizing a task similar to the task that the author of the new program wishes to solve. This does not require further explanation, except noting that in order to effectively modify programs, access to their source codes is necessary. Source code is a human-readable expression of a program; thus, the practice of distributing software in object code, allowing only for its execution on computers prevents modifications. Therefore, public availability of the source code is a condition sine-qua-non of a program being free software.

Programs, expressed in either source or object code usually form a part of a bigger whole, referred to as „software”. They form it together with: (1) software development tools; (2) preparatory material; (3) input data; (4) program's output; (5) additional material (documentation, help modules) and (6) interfaces (including graphical „look and feel”).<sup>16</sup> Moreover, program as such may be considered to contain textual and non-textual elements. The former are the source or object code, while the latter are, for example, the program's structure (so called SSO – structure, sequence and organization), flow charts, menu structure, or interfaces. Thus, the use of another person's program may not necessarily be accomplished by copying textual elements, such as portions of source code. Since the value and usefulness of the program lies mostly in its non-textual elements, they are often copied too.

There are numerous common tasks to solve while programming. Besides, most software has to perform functions other than the main one. In order to avoid the necessity of writing the same again and again for every program, so-called software libraries are available. A library is not a program on its own, it only contains a set of useful variables, data, functions, procedures, or bigger modules (subprograms). The use of libraries in programs is referred to as „linking” and may be static or dynamic.

Static linking occurs when the program and the library are joined together in one file. It is done at the moment of translation, when the human-readable source code is changed into a computer-executable object code file. The program created in such way is then distributed together with the library linked to it. Further in the article, the term „static linking” will be used to mean any other distribution of programs joint in one file, even if neither of them is a library.

---

<sup>15</sup> Eric S. Raymond, *The Cathedral and The Bazaar*, <http://www.catb.org/~esr/writings/cathedral-bazaar/>.

<sup>16</sup> Cf. DAVID BAINBRIDGE, *SOFTWARE COPYRIGHT LAW*, 1-3 (Butterworths, London, Edinburgh, Dublin, 4th ed., 1999).

Dynamic linking is a more popular software writing technique, but at the same time leads to a more complicated legal situation. Program file contains only references, which result in the use of necessary libraries by the computer either in the time when the program is loaded into the memory (loadtime linking) or even at the moment of its execution (runtime linking). Dynamic linking allows not to distribute the libraries together with the program; they are available directly on the user's computer and form a part of the operating system used to run the program. Additionally, to dynamically link a library does not necessarily cause it to be used by the program, since the final „decision” on this is taken by the user's computer, not the programmer. It may thus happen, that the dynamic library finally linked would be a different one than the one chosen by the program's author.

For the purposes of this article, the term „dynamic linking” will be used to cover other subtle means of using programs by other programs. For example, a www server, such as Apache Web Server may be equipped in a dynamically loadable module containing a scripting language, such as PHP. Moreover, programs may communicate between each other using publicly available sets of interfaces, („application program interface”, API). An interesting example of such interaction is Wine program, which emulates the public set of procedures of Windows and allows for the execution of programs written for this system under a Linux machine. Programs may also cooperate on the level of communication protocols. One can name here the program Samba, which allows to link a Linux/Unix machine to a Windows-operated network using its SMB/NMB protocols.

It is not possible to describe here the software writing techniques in more detail. The reader; however, should realize that the information above is a generalization. A number of computer languages and operation systems are available, and each of them allows or requires the use of certain different techniques.

Before attempting to analyze the GPL in a strict legal sense, it seems a good idea to try to classify the effects of using the techniques described above to one of the four categories proposed by Vetter.<sup>17</sup> Generally speaking, then, a simple modification or copying of code results in the creation of a program belonging to the first category. Here belongs also the copying of copyright-protected non-textual elements. Static and dynamic linking seems to belong to the second category in most cases. The remaining techniques should be classified in the category three, with the reservation, that a highly subtle relation between programs should result in a „mere aggregation” and consequently the exclusion from „copyleft” under the GPL alone. It shows that the borderline set using the classification above is blurred; thus, it is necessary to clarify it further by searching for the meaning of the license terms as provided by the current law.

## **5. Scope of copyleft under American law**

GPL uses American legal language and directly references to institutes codified in Copyright

---

<sup>17</sup> See text accompanying FN. 4.

Act.<sup>18</sup> When analyzing „copyleft” clause the following definitions of Sec. 101 should be kept in mind:

A „**collective work**” is a work, such as a periodical issue, anthology, or encyclopedia, in which a number of contributions, constituting separate and independent works in themselves, are assembled into a collective whole.

A „**derivative work**” is a work based upon one or more preexisting works, such as a translation, musical arrangement, dramatization, fictionalization, motion picture version, sound recording, art reproduction, abridgment, condensation, or any other form in which a work may be recast, transformed, or adapted. A work consisting of editorial revisions, annotations, elaborations, or other modifications, which, as a whole, represent an original work of authorship, is a „derivative work”.

The point of departure should be to specify the meaning of a „derivative work” and „collective work” as applied to computer programs. A derivative work is undoubtedly created when copyright-protected elements of program are copied. Copying of textual elements of a program may be easily found by comparing the texts of programs. In order to estimate whether any non-textual elements were used and if it falls under the copyright scope, American courts developed a number of tests, such as the most well-known Abstraction-Filtration-Comparison Test (hereafter „AFC”) from *Computer Associates, Inc. v. Altai, Inc.*<sup>19</sup> It is based on a founding principle of copyright law, which grants protection to expression of ideas, not the ideas themselves.

In the first phase of AFC the court divides the program into its levels of abstraction, the examples of which were pointed out by Ravicher: the program's main aim, its architecture, abstract data types, algorithms, data structures, source code and object code.<sup>20</sup> In the next phase, each part of the program found in phase one is filtered in the search of copyright-protectable material. The following items are left out: (1) ideas, (2) expression incidental to the ideas resulting in merging the idea and its expression into one (merger doctrine), and (3) expression dictated by external conditions, such as for example compatibility requirements (scenes a faire). Additionally, the (4) public domain elements and (5) non-original elements are filtered away. In the last phase, a comparison between protected elements found in the preceding phases with the other program; thus, it is estimated if they were copied. Not only the fact that they were copied, but their relative importance as compared to the whole program is taken under consideration.

AFC is believed to be one of the strictest tests used to determine which elements of a program deserve protection. It rarely happens that the protection for any type of code is not granted; however, AFC often leads to allowing to copy non-textual elements. All of the elements that are not

---

<sup>18</sup> Copyright Act of the United States of America, 17 U.S.C. Sec. 101-1332 (1976, as amended).

<sup>19</sup> 982 F 2d 693, 706 (2nd Cir 1992).

<sup>20</sup> Dan Ravicher, *Software Derivative Work: A Circuit Dependent Determination*, 3 [http://www.pbwt.com/Attorney/files/ravicher\\_1.pdf](http://www.pbwt.com/Attorney/files/ravicher_1.pdf).

copyright-protected may be used in new programs without limitations; consequently, they cannot fall under the scope of „copyleft”. It should be concluded that compared to copying of non-textual elements, a creative modification of a program leads to the creation of a derivative work with a far greater probability.<sup>21</sup>

American copyright law grants the author the monopoly for the creation of derivative works.<sup>22</sup> A derivative work is protected by copyright independently from the original, the protection; however, does not extend for the works where the protected material was used illegally.<sup>23</sup> Thus, if a derivative work was created without the original author's authorization or outside the fair use exception, it is not protected by copyright („infringing derivative”). American legal doctrine agrees that the copyright in such derivative does not vest neither in the original author, nor its creator, but fails to produce a positive answer on its legal status. Lemley believes that such derivative works fall into public domain.<sup>24</sup> If this is the case, infringing derivatives would not only escape „copyleft”, but also provide a way of appropriating a free program using the mechanism described before. It should be pointed out; however, that the probability of creating an infringing derivative of a free program is low, since the GPL allows for any modification.<sup>25</sup>

Attempts to determine the exact scope of „copyleft” get even more complicated since there exist code reuse techniques far more advanced than mere copying or modifying. The general procedure used by the American doctrine is to estimate whether they lead to the creation of derivative works or at least collective works.

Static library linking described above is usually considered to effect in a collective work. For the distribution of such works, as for the distribution of derivative works, the original author's authorization is necessary.<sup>26</sup> Thus it seems that static linking to a free program results in the obligations described in the „copyleft” clause to come into force. On the other hand, the GPL treats „mere aggregation” as outside of its scope. Free Software Foundation believes that mere aggregation occurs if programs are just distributed together on one medium, while joining them in one executable file definitely falls under „copyleft”.<sup>27</sup> These; however, are extreme examples with a

---

<sup>21</sup> Some model free software licenses limit the scope of their „copyleft” to modifications only. Cf. Mozilla Public License, v. 1.1, <http://www.mozilla.org/MPL/MPL-1.1.html>.

<sup>22</sup> 17 U.S.C. Sec. 106.

<sup>23</sup> 17 U.S.C. Sec. 103(a).

<sup>24</sup> Mark A. Lemley, *The Economics of Improvement in Intellectual Property Law*, 75 TEXAS LAW REVIEW 989, 1022 (1997).

<sup>25</sup> There is an extensive jurisprudence on the validity and enforceability of „shrink-wrap” and „click-wrap” licenses (to which the GPL may be compared), from the point of view of contract law. Thus, it may happen that the license would not bind the parties in certain cases (*Step-Saver Data Sys., Inc. v. Wyse Tech.* 939 F.2d 91 (3d Cir. 1991), *Specht v. Netscape Communications Corp.* 306 F.3d 17 (2nd Cir. (N.Y.) 2002), Kevin W. Grierson, *Enforceability of "Clickwrap" or "Shrinkwrap" Agreements Common in Computer Software, Hardware, and Internet Transactions*, 106 AMERICAN LAW REPORTS 5th 309). As the consequence, given no legal authorization for the preparation of derivative works, the resulting works would be infringing derivatives.

<sup>26</sup> LEE A. HOLLAR, LEGAL PROTECTION OF DIGITAL INFORMATION, VI.D.4, <http://digital-law-online.info/lpdi1.0/treatise27.html>

<sup>27</sup> Free Software Foundation, *Frequently Asked Questions about the GNU GPL*,

lot of middle-ground situations possible.

In case of dynamically linked libraries, they are not distributed together with the program. Access to the libraries is provided by the user and some authors believe that it is him who has to secure any rights necessary for their use.<sup>28</sup> In such a situation, the author of a program dynamically linked to a free program would not have any obligations resulting from „copyleft“. Moreover, it is doubtful, whether the users of such program would have these obligations, since being not the rights holders, they would have no possibility to fulfill them.

The American doctrine has not agreed whether dynamic linking leads to the creation of a derivative work of a program. Rosen believes that dynamic linking is just a temporary relation between programs and does not lead to the modification of the linked program.<sup>29</sup> This reasoning; however, is based on the definition of a derivative work as a modification of the original work, one way or another. Free Software Foundation considers both static and dynamic linking as the creation of joint works; thus, subject to the scope of „copyleft“ clause.<sup>30</sup> This position, expressed in the GPL FAQ published by this organization should have an important influence on the interpretation of this license, but it should be noted that it is excessive.<sup>31</sup> Free Software Foundation itself admits that the final answer to these controversies should be left to the courts, which should evaluate not only the mechanism, or mode of communication between programs, but also its content (the type of information exchanged).<sup>32</sup>

One of the crucial programs available under the GPL is the Linux kernel. Since it forms the basis for any GNU/Linux operating system, then everyone writing software for this system should ponder whether their works fall under the scope of „copyleft“. Linus Torvalds, apart from expressing his opinion in numerous occasions,<sup>33</sup> decided to release Linux under a following note on top the GPL:

```
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
```

---

<http://www.fsf.org/licensing/licenses/gpl-faq.html#MereAggregation>

<sup>28</sup> HOLLAAR, FN. 26.

<sup>29</sup> Lawrence Rosen, *The Unreasonable Fear of Infection*, 2, <http://rosenlaw.com/html/GPL.PDF>.

<sup>30</sup> Free Software Foundation, *Frequently Asked Questions about the GNU GPL*, [http://www.fsf.org/licensing/licenses/gpl-faq.html#IfLibraryIsGPL et seq](http://www.fsf.org/licensing/licenses/gpl-faq.html#IfLibraryIsGPL_et_seq). For these, who would like to allow linking with proprietary programs, the FSF designed the Lesser General Public License (LGPL, available at: <http://www.fsf.org/licensing/licenses/lgpl.txt>). There is a question then, whether in an individual case, the choice of the GPL instead of the LGPL should not be considered as the parties' will to cover by the „copyleft“ both static and dynamic linking.

<sup>31</sup> Adopting a similar position in case of the dynamically linked libraries available with the Windows operating system would lead to the total monopolization of the application market for this system by Microsoft, by giving it the control over the distribution of the programs using these libraries.

<sup>32</sup> Free Software Foundation, *Frequently Asked Questions about the GNU GPL*, <http://www.fsf.org/licensing/licenses/gpl-faq.html#MereAggregation>

<sup>33</sup> See e.g. [http://www.kerneltraffic.org/kernel-traffic/kt20031226\\_246.html#1](http://www.kerneltraffic.org/kernel-traffic/kt20031226_246.html#1) or [http://www.kerneltraffic.org/kernel-traffic/kt20021021\\_189.html#32](http://www.kerneltraffic.org/kernel-traffic/kt20021021_189.html#32).

heading of "derived work". ...<sup>34</sup>

Certainly it is right that common programs written for the use on a computer running Linux-based operating system, belonging to so-called „user area” are not covered by „copyleft”. It follows for example from the rule that excludes from copyright protection expressions dictated by external conditions or can additionally be excused by the fair use of operating system. However, when trying to specify when the operating system ends and user area begins, the informatics has similar difficulties as the theory of law, when it tries to differentiate derivative works from separate and independent works.

## **6. Determinants of copyleft under Polish law**

### **A. General remarks**

There is a question, whether the scope of „copyleft” clause should be construed in accordance with the scope of copyright protection, or can it be extended further. It may prove practical to answer this question, especially in case when the analysis of the GPL leads to the conclusion that it indeed attempts to reach outside the copyright monopoly. For example, if it should require that the source code is published of the programs merely inspired by a GPL-licensed program. In such a case it would be necessary to ponder, if such construction of its scope is valid and enforceable, and on what ground.

It is a common practice to put into licenses the limitations having its source not in copyright, but rather in contract law, especially in case of proprietary software. Such arrangements are limited under American law, for example by the „preemption doctrine” flowing from the federal structure of law, which does not allow the courts to enforce state contract law in a way that it refers to the matters already covered by federal copyright law. But the practical effect of such limitations is weak.<sup>35</sup>

In any case, even a glimpse at Polish law would probably allow to point analogical limitations for the use of contractual freedom to reach out of copyright protection. It may prove useful, in this context, to analyze the proposal of the doctrine to construct GPL as an individual act, not a contract. It is often used by the FSF as an argument for the license's validity and enforceability, but obviously it limits the possibilities to extend its scope over the reach of copyright.

---

<sup>34</sup> Linus Torvalds, *Note to Linux Kernel*, <http://www.linux.de/linux/gnu.html>. It is a very important guideline, especially considering the ease of its use, since the normal system calls of the Linux kernel are published (Vetter, FN. 11, 116, quoting the email of Linus Torvalds available at: <http://www.atnf.csiro.au/people/rgooch/linux/docs/licensing.txt>). There is a question, whether the note by Linus Torvalds is a proof of market practice that considers any call of a program by another program (dynamic linking for the purposes of this article) as the creation of a derivative work.

<sup>35</sup> Cf. Lorin Brennan, *Why Article 2 Cannot Apply to Software Transactions*, 38 DUQUESNE LAW REVIEW, 459 (2000) (analyzing, inter alia, incompatibilities between UCC Art. 2 (Sale of goods) and Copyright Act); Mark A. Lemley, *Beyond Preemption: The Law and Policy of Intellectual Property Licensing*, 87 CALIFORNIA LAW REVIEW, 111, 113 (1999) (pointing to, apart from the “preemption doctrine”, the “copyright misuse” as the source of license limitations). See also: Copyright Act. Sec. 301(a) and U.S. Const., Art. VI, Sec. 2 (Supremacy Clause), and Dennis S. Karjala, *Federal Preemption of Shrinkwrap and On-Line Licenses*, 22 UNIVERSITY OF DAYTON LAW REVIEW 511, 527-528 (1997) (pointing out that the adhesive form of licenses calls for treating the resulting claims as based in copyright, not contract).

These issues will not; however, be developed in this article, because we assume here that the scope of „copyleft” cannot be wider than the copyright law protection of computer programs. Even if it were legally possible to draft the licensees' rights and obligations differently, it seems that the literal reading of the GPL as well as the underlying ideas of the Free Software Movement, do not constitute the attempt to extend copyright monopolies, but just the innovative use of author's rights. Moreover, the terms used in the license, such as „derivative works” or „collective works” which determine the scope of a „work based on the Program” belong directly to copyright law and it seems not necessary to attribute to them the meaning different than the one already given by this law.<sup>36</sup> In other words, if, for example, dynamic linking should fall under the scope of „copyleft”, it will not be considered the effect of an expansive construction of the license terms, but rather the effect of copyright institutes' interpretation that the license refers to.

### **B. Idea-expression dichotomy**

The limit of copyright protection is set by Art. 1.2<sup>1</sup> of Polish Copyright Law,<sup>37</sup> which provides that the protection applies only to expression but not to discoveries, ideas, procedures, methods or rules of operation, as well as mathematical concepts. Additionally, in relation to computer programs, Art. 74.2 declares:

The protection granted to computer program extends to every form of its expression. Ideas and rules forming the basis of any of the elements of a computer program, including the basis for the interfaces, are not protected.

It follows that Polish Copyright Law observes the same basic copyright rule which was used by the American courts in decisions such as *Altai*.<sup>38</sup> It does not immediately follow; however, that the way to determine which elements are protected would be to apply the AFC. The interpretation of Polish provisions is subject to the position of the doctrine on the scope of terms such as „content” and „form” of a work, or the way works are believed to be constructed (compare monolithic concept of Bleszyński<sup>39</sup> with a three-layer concept of Kopff<sup>40</sup>).

Nowicka believes, that the *lex specialis* of Art. 74.2 releases from the obligation to differentiate the form from the content in case of computer programs.<sup>41</sup> According to the literal

<sup>36</sup> Compare Vetter, FN. 11, 111 (pointing out that the parts of the GPL describing the scope of „copyleft” permit the activities already permitted by the copyright law) with Robert W. Gomulkiewicz, *De-bugging Open Source Software Licensing*, 64 UNIVERSITY OF PITTSBURGH LAW REVIEW 75, 90 (2002) (claiming that the GPL contains the definition of „derivative works” wider than the one in the Copyright Act; thus it attempts to cover, e.g. works based on unprotected ideas or data).

<sup>37</sup> Ustawa z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych. (Dz. U. 1994 Nr 24 poz. 83, z późn. zm.).  
<sup>38</sup> See FN. 19.

<sup>39</sup> JAN BLESZYŃSKI, TŁUMACZENIE I JEGO TWÓRCA W POLSKIM PRAWIE AUTORSKIM [TRANSLATION AND ITS CREATOR UNDER POLISH COPYRIGHT LAW], (Warszawa 1973).

<sup>40</sup> A. Kopff, *Wpływ postępu techniki na prawo autorskie [Influence of technical progress on copyright law]*, w: 48 PRACE Z WYNAŁAZCZOŚCI I OCHRONY WŁASNOŚCI INTELEKTUALNEJ [WORKS ON INVENTIONS AND PROTECTION OF INTELLECTUAL PROPERTY] 104 (1988).

<sup>41</sup> AURELIA NOWICKA, PRAWNOAUTORSKA I PATENTOWA OCHRONA PROGRAMÓW KOMPUTEROWYCH [COPYRIGHT AND PATENT PROTECTION FOR COMPTUER PROGRAMS, 116-117 (Dom Wydawniczy ABC 1995).

wording of the provision, she declares „ideas” and „rules” as not protected. She understands that the protected „every form of its expression” are the same as „forms of expression” and similar to „modes of expression” as specified by the general rule of Art. 1.1. Moreover, Golat believes that the exemption of „ideas” and „rules” does not necessarily allow to treat these elements as equal to the content layer of the program and does not allow to exclude every content from copyright protection.<sup>42</sup>

This leads to a generally accepted conclusion, that the protection applies to any form of the program's code. What remains to be clarified, is the scope of the protection for non-textual elements. Golat believes, that not only general, but also more specific ideas of a program are not protected.<sup>43</sup> Moreover, he distinguishes mathematical algorithms, that is the methods of solving a task expressed in the form of mathematical operations, that are discovered, not created,<sup>44</sup> from „informatics' algorithms”, defined by him as: „logically structured sets of instructions, the realization of which according to the respective rules allows to solve specific tasks existing within the target task forming the basis for the preparation of the computer program.”<sup>45</sup> According to this distinction, mathematical algorithm would constitute the unprotected content of the informatics' algorithm, while the protected form of this expression would be the graphical or descriptive presentation of how the mathematical algorithm is applied. This form, is the same as „program's structure, expressed in the way the set of instructions is crafted, [but not the unprotected logical-mathematical idea for the solution of the task]”.<sup>46</sup>

In the attempt to sum up these speculations, one may repeat after Nowicka, that the search for the limits of copyright protection for computer programs in Polish law may lead to similar results, as the AFC in the U.S.<sup>47</sup> Specifically, the distinction between mathematical and informatics' algorithm proposed by Golat, may be compared to the search for the levels of abstraction by the court in *Altai*. It is not clear; however, if under Polish law, the filtration of protected elements would lead to same results as the AFC phase two, since it requires the use of merger and scenes a faire doctrines, not recognized by Polish law.

It should also be kept in mind that *Altai* is the law in 2<sup>nd</sup> Circuit, and can provide persuasive authority in other circuits; while the Polish doctrine is not binding the courts at all and the written law is not specific enough. It means that there is a considerably higher risk under Polish law connected with the exact scope of copyright protection for computer programs, which is logically followed by similar uncertainty of the scope of „copyleft”.

---

<sup>42</sup> KATARZYNA GOLAT I RAFAŁ GOLAT, PRAWO KOMPUTEROWE (ZAGADNIENIA PODSTAWOWE) [COMPUTER LAW (BASIC ISSUES)], 41 (Wydawnictwo Prawnicze, Warszawa 1998)

<sup>43</sup> *Id.* 43.

<sup>44</sup> *Id.* 47.

<sup>45</sup> *Id.* 45.

<sup>46</sup> NOWICKA, FN. 41, 124.

<sup>47</sup> *Id.* 125.

### C. Fair use

The „copyleft” clause does not cover the elements of program not protected by copyright. Furthermore, it may not be applied to such protected elements, the use of which is allowed under the provisions on fair use.

The scope of fair use as applied to computer programs relevant for the purposes of avoiding „copyleft” clause seems narrow. Lex specialis of Art. 77 excludes the application to computer program of most of the fair use provisions of Copyright Law. Nevertheless, outside this exclusion remains Art. 29, which regulates the fair use quotations. Golat believes that this provision allows to reference to other programs in a newly created program.<sup>48</sup> Unfortunately, Polish doctrine does not provide additional guidance on the limits of such referencing. In particular, to the best knowledge of the author of this paper, no analysis of the aforescribed code reuse techniques has been made in the context of fair use quoting.

Art. 29 allows to quote fragments of already published works or small works in whole, if it may be explained by, *inter alia*, „laws of the type of arts”. One should consider whether the laws of the art of software writing would be such to permit the use of ready-made libraries or modules in new programs. Perhaps it should also be worth considering, whether the Free Software Movement, with its specific moral norms, customs and the interpretation of the GPL advocated by its drafters forms a type of art on its own. These laws would permit quoting to a great extent; however, under condition of observing the „copyleft” clause. On the other hand it is not clear whether the „copyleft” construed as the law of art may reach as far as requiring the publication of source codes from anyone who included in his program even the smallest portion of free code.

Practical importance of fair use quotation lies in citing fragments translated from source to object code that is not human-readable. If there is such legal possibility, it undoubtedly is the way to appropriate fragments of free programs or even small free programs in whole. It is well known that to quote is to take somebody's work as it is, without changing a word, but at the same time it is allowed to cite a translation,<sup>49</sup> while the object code is nothing else than a translation of the source code. The requirement to mention name of the creator or source provided for in Art. 34 may be an obstacle here, but it does not seem to totally block the possibility to quote programs covered by the GPL. However, the relation between Arts. 29 and 35 should definitely be considered, that is whether the avoidance of „copyleft” clause with the use of fair use quotation infringes the „due interest” of free software writers.

Nevertheless, it should not be expected that Art. 29 provides a real threat to the „copyleft” clause, not to mention the possibility to use somebody else's code in one's own programs at all.

---

<sup>48</sup> GOLAT, FN. 42, 144.

<sup>49</sup> J. BARTA, M. CZAJKOWSKA-DĄBROWSKA, Z. ĆWIAKALSKI, R. MARKIEWICZ, E. TRAPLE, USTAWA O PRAWIE AUTORSKIM I PRAWACH POKREWNYCH. KOMENTARZ [COPYRIGHT LAW. COMMENTARY], tezy do art. 29 (Dom Wydawniczy ABC, 2001, wyd. II)

Besides the limitations contained in the provision itself, it should be stressed, that fair use is a limitation on a general rule of author's monopoly and should not be interpreted excessively (*exceptiones non sunt extendendae*). Moreover, special provisions for computer programs impose even farther limitations, especially for the translation of computer programs, which is necessary to avoid the GPL. Namely, according to Art. 75.2.3 it is allowed only in order to reach interoperability between independently created computer program with other computer programs, which can rarely be an excuse for the creation of a proprietary program containing portions of free code.

#### **D. Derivative works**

Derivative works referred to by the GPL are covered by Art. 2 of Polish Copyright Law. Part of Polish doctrine took the position that setting the scope of copyright protection for computer programs in Art. 74.4.2 constitutes *lex specialis* in relation to the provisions on derivative works. Even though this position seem well-grounded, it does not seem to follow that the activities listed in Art. 74, that is the translation, adaptation, change of structure or any other changes in computer program, may not be covered by the „copyleft” clause. Reference to derivative works institution made in the fragment of the GPL quoted above provides just interpretation guidance, while the express provisions of this license include in the scope of „copyleft” works in whole or in part containing or derived from the program or any part thereof. Activities listed in art. 74.4.2 effect in the creation of such works.

Byrska believes, that these activities may be performed by persons other than the copyright holder in the original, but the copyrights to new works created in this way vest in him, not them.<sup>50</sup> At the same time, the American law discussed above considers the creator of a derivative work to hold copyrights in it; while infringing derivatives are believed by some authors to fall into the public domain.<sup>51</sup> Similar position in Polish literature is taken by Nowicka, who believes that unauthorized derivatives of a computer program are not subject to copyright law.<sup>52</sup> Definitely, the most beneficial for the Free Software Movement would be to grant the rights in the modified program directly to the author of the original. Then, he would be able to directly require others to observe four freedoms of Stallman, without the need to call upon the license. However, the Polish doctrine has not reach a common ground on this issue.<sup>53</sup>

It should be separately analyzed, how Polish law treats other techniques of code reuse, not just simple copying or modification. Depending on the approach; it should be considered whether they form a derivative work of a program, or, in case of finding that Art. 2 does not apply, whether

---

<sup>50</sup> Małgorzata Byrska, *Prawne aspekty modyfikowania programu komputerowego [Legal aspects of modifying computer program]*, 4 KWARTALNIK PRAWA PRYWATNEGO [PRIVATE LAW QUARTERLY], 693, 715 (1996).

<sup>51</sup> See text accompanying FN. 24.

<sup>52</sup> NOWICKA, FN. 41, s. 134.

<sup>53</sup> Cf. e.g. BARTA ET. AL., FN. 49, comment 23 to Art. 74 „There is definitely no ground for the interpretation that art. 74 grants the rights to the derivatives of computer program to the author of the original” (translated by the author).

they result in a translation, adaptation, change of structure or any other change in a computer program. It should be noted, that the second option logically allows to use the approach of Rosen already presented before, who believes dynamic linking is outside the scope of „copyleft”, since it results in no change in linked programs.<sup>54</sup>

An interesting approach is to treat the use of free software in new programs as the continuation of works. Traple distinguishes between the continuation of a not finished work from the continuation in the sense of writing the follow-up to an already finished work.<sup>55</sup> Here, a special feature of software industry should be pointed out, which is that the aim is not to produce any finished, closed work. Actually, the rule is to constantly develop software in order to adjust them to the changing environment, that is for example the technical progress or the users' needs. The users of these special types of works, computer programs, also do not expect to be provided with a final usable product. This is because it is impossible to predict every state in a computer running given software, which results in every program containing errors („bugs”). It is widely accepted that new versions are created („upgrades”, „updates”, „patches”, „service packs”) which are intended to eradicate these errors.<sup>56</sup>

In the light of the above, it seems reasonable to treat the majority of software as not finished works. If so, then writing a portion of code in order to correct errors in such software should lead to the creation of a derivative work of it.<sup>57</sup> Consequently, the use of a library or module in a new program should be considered a continuation of a closed work, which may be considered a derivative work in case original elements of such library or module were used.

Doubts arise since there is no actual merger in a new program of the free program used in it in case of dynamic linking. Patches are also usually available without the original flawed program; they only contain corrected lines of code with the information where they should be put in the patched program. It means that the original program may not be recognizable in the program created as the patch or that is dynamically linked to it. In such a case, according to Traple, no derivative work is created.<sup>58</sup>

### ***E. Collections, collective works, joint works***

Instead of treating the effects of code reuse as derivative works or, alternatively, as covered

<sup>54</sup> See FN. 29.

<sup>55</sup> BARTA ET. AL, FN. 49, comment 29 to Art. 2.

<sup>56</sup> Advocates of Free Software Movement raise that the abovedescribed feature of software production makes the public availability and the grant of modification right a perfect mean of assuring faster and more effective bugs correction. It is hard not to agree with this claim, but its proof is outside the scope of this article.

<sup>57</sup> BARTA ET. AL, FN. 49, comment 29 to Art. 2. Certainly, depending on the facts it is possible that there a work of joint authorship is created. This; however, is outside the scope of this article, since it attempts to estimate the scope of „copyleft” as constructed against the licensee. Thus, we assume that there is no agreement between the parties as to the joint creation of a program, even though such situations are extremely popular in the Free Software Movement.

<sup>58</sup> BARTA ET. AL, FN. 49, comment 30 to Art. 2 (criticizing the decision of the Supreme Court, SN IK, 26 November 1930, OSP 1931 nr 10 poz. 87, where the court copyright infringement in the preparation of the solutions to previously published high school Latin tests, since the solutions were considered the translation, that is a derivative work of the tests.

by Art. 74.4.2, they should be considered to fall under provisions regulating collections (art. 3), collective works (art. 11) or works joint for distribution (art. 10). Since the last option expressly contemplates an agreement between authors of joint programs, it will not be considered in this article, as well as the joint authorship of programs (see explanations in FN. 57).

As far as the other options are taken under consideration, it should be concluded that they fall under „copyleft” clause due to the fact that the authors of works used in collections or collective works retain their rights. Certainly, there are doubts in relation to dynamic linking where there is no actual coexistence of programs in one work. Nevertheless, in case of static linking, both the retainment of rights in Art. 3 as well as the rights to respective parts provided for in Art. 11 allows their holders to formulate conditions for the use of their works. Such conditions may affect the way how the whole is distributed, and the „copyleft” clause is one of them.

Here, it should be expressly repeated after many free software advocates that the obligations from the „copyleft” clause, in particular the obligation to publish source codes, are activated only when the programs derived from a GPL-ed program are distributed. Thus, the use of programs linked one way or another to free software and keeping secret their source codes is completely in-line with the GPL.

As it was already risen here (see text accompanying FN. 27) the scope of „copyleft” clause does not cover the mere aggregation of programs for joint distribution. Thus, also under Polish law there is the need to distinguish between mere aggregation and the creation of a collection or collective work. It seems that an important guidance is provided by the requirement of creative contribution for the arrangement, structure or setting of the works in a collection, provided for by Art. 3. The creative contribution, as applied to computer programs might consist in a new functionality of the collection, lacking in programs existing separately, when each is performing its task alone, without any interaction with the others.

## **7. Conclusion**

The analysis above does not result in a clear answer to the question on the scope of „copyleft” under Polish law and it seems that its delimitation is simply impossible. The reasons are complex. Firstly, this is the result of general language of the GPL, whose drafters intended to provide something like a model law for free software, instead of a particular licensing agreement. Secondly, it is because the factual states occurring in the software industry are complex and constantly change due to its progress.

These determinants are accompanied by the third one, that is the provisions, jurisprudence and doctrine of copyright law. The analysis of the „copyleft” clause in this context brings about the old issues such as idea-expression dichotomy, the scope of fair use or the difference between inspired but independent works and derivative works. Certainly, in a given situation, these would

have to be somehow applied; and judging from the decisions such as *Altai*, the scope of copyright protection is drawn in a coherent way, with socio-economic sense.

To sum up, it is worthy to stress that the difficulties in estimating the scope of the „copyleft” clause arise only on the borderline between copyright and public domain. In more typical situations, when an apparent copying or modification occurred, the conclusion that „copyleft” applies should be possible to draw more easily. With some reservations one can formulate the position that static linking is also covered by the „copyleft” clause. However, attempts to extend it on dynamic linking should be considered quite risky. Thus, the scope of „copyleft” clause in Polish and American law seems similar.

It should be kept in mind, that the „copyleft” clause has not been drafted to catch an unwary licensee, but is thoroughly pro-social. Its intent is to secure the interests of persons active in the creation of free software and not to allow to monopolize the effects of their work by unfair free-riders, whose activities might result in limiting the public right to access the software. Thus, the obligations of the „copyleft” clause end where the rights of other people start, which is expressed in the following words of the GPL: „it is not the intent of this section to claim rights or contest your rights to work written entirely by you.”<sup>59</sup>

---

<sup>59</sup> See FN. 9.