A Kantian approach to Free Software

Benedikt Peetz
benedikt.peetz@b-peetz.de

Supervisor: Ms. Herrmannsdörfer

Wednesday 20th March 2024

Contents

I am grateful for the valuable insights and suggestions provided by Silas Schöffel. I am also thankful for the work on the large language model MixTral-8x7B by Mistral AI which helped me with some formulations.

1 Introduction

This essay is Free Software, its source code can be found at https://codeberg.org/bpeetz/kant_and_free_software.

1.1 Motivation

The argumentation for Free Software has adhered to deontological ethics since its inception, as demonstrated by the Four Freedoms outlined in subsection 1.2—rights that must be categorically granted to all users.

This deontological influence is prevalent in every one of the Four Freedoms: For example, Freedom 2, outlining the right to study the program to see what it does and to transparently decide whether to use the software, emphasizes the importance of the *autonomy* of a person.

But hitherto, Immanuel Kant, who has modernized the concept of deontological ethics and introduced the concept of autonomy, has not been mentioned by Richard Stallman, to my knowledge. Consequently, I will try to address that, by connecting Kantian ideas (like Kant's Categorical Imperative (CI)) directly to the ideals of Free Software.

1.2 Definitions

As this essay will deal with the ethics of Free Software, the terms around Free Software should be defined:

Source Code Source code is defined as the preferred form of the program for making changes in. Thus, whatever form a developer changes to develop the program is the source code of that developer's version[1].

Software computer programs, procedures, and possibly associated documentation and data pertaining to the operation of a computer system[2].

Software Library a software library containing computer readable and human readable information relevant to a software development effort[3].

Free Software Software, which adheres to the four essential freedoms[4]. These are:

0. The freedom to run the program as you wish, for any purpose [. . .].
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.
2. The freedom to redistribute copies so you can help others [. . .].
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.

Proprietary/nonfree Software Software not following the definition of free software (see subsection 1.2)[5].

[1]GNU-Project (ed.): What is Free Software?, Version: 1.169, Feb. 2021, URL: https://www.gnu.org/philosophy/free-sw.html.en#make-changes (visited on 20/03/2024).

[2]ISO/IEC/IEEE International Standard - Systems and software engineering–Vocabulary, in: ISO/IEC/IEEE 24765:2017(E), Aug. 2017, 3.2741, DOI: 10.1109/IEEESTD.2017.8016712, p. 329.

[3]Ibid., p. 332.

[4]GNU-Project (ed.): What is Free Software?, Version: 1.169, Feb. 2021, URL: https://www.gnu.org/philosophy/free-sw.html.en#four-freedoms (visited on 14/02/2024).

[5]Idem (ed.): Categories of Free and Non-

1.3 Philosophical Premise

1.3.1 Kant

As I have mentioned in subsection 1.1 on the preceding page, I intend to apply the Kantian ethics to examine the morality of Proprietary Software development. This requires the knowledge about *how* Kantian ethics can be used to evaluate the morality of an action.

Firstly[6], one needs to identify the maxims behind the action being evaluated. This could be something like: "It is acceptable to write Proprietary Software".

Secondly[7], one needs to, following the CI, decide, whether an action can be universalized without causing contradictions. Kant writes—in German—that "dies [. . .] der Kanon der moralischen Beurtheilung derselben überhaupt [ist]"[8]

This last step is effectively about applying the CI to the moral question, leaving immoral or self-interest driven actions to being exposed as not universalizable—as these actions would contradict themselves when doing so.

Kant splits these contradictions into two categories: A contradiction in the conception itself: the maxim becomes useless when universalized. And secondly, into a contradiction in will, where it would be impossible to want to universalize the law, as such a desire ("Wille") would contradict itself[9].

free Software, URL: https://www.gnu.org/philosophy/categories.html#non-freeSoftware (visited on 16/03/2024).

[6]Cf. Immanuel Kant: Grundlegung zur Metaphysik der Sitten, AA IV, 1785, URL: http://kant.korpora.org/Band4/421.html, pp. 421 – 424.

[7]Cf. ibid., p. 424.

[8]Ibid.

[9]Cf. ibid.

1.3.2 Kant's Categorical Imperative

The aforementioned test for the morality of an action uses the CI, of which Kant has formulated multiple ones, each one of them focusing on a slightly different area and addressing different critiques.

Notably, Kant also writes that the three ways of formulating the CI are "objectiv=praktisch" not different[10].

I will be working with the "Kingdom of Ends" CI, as it highlights the importance of thinking of the different maxims in a *system* and not only as individual decisions[11]:

> Demnach muß ein jedes vernünftige Wesen so handeln, als ob es durch seine Maximen jederzeit ein gesetzgebendes Glied im allgemeinen Reiche der Zwecke wäre.

Additionally, this CI still keeps the requirements of the other two CIs intact, by defining the "Kingdom of Ends" ("Reich der Zwecke") as a "systematische Verbindung vernünftiger Wesen durch gemeinschaftliche objective Gesetze"[12]. Therefore, every being composing this Kingdom, is also subjected to laws, which it has put onto itself and vice-versa with the other beings of said Kingdom.

[10]Cf. ibid., p. 436.

[11]Ibid., p. 438.

[12]Ibid., p. 433.

2 Examination of morality

This essay is about whether the development of Proprietary Software is morally acceptable. Applying the steps outlined in subsubsection 1.3.1 on the previous page means that we must start with trying to encompass the maxims behind the action.

Deriving from the definition of Proprietary Software it is to be concluded that there are four reasons to not develop Free Software, as the developer does apparently not care about providing the four freedoms to their users. Thus, a possible maxim could be: “I do not want my users to follow one of the four freedoms”.

To be able to be more precise—and because access to the source code is a precondition to two freedoms—, I am going to use the following maxim for the further evaluation: “I wish, that the source code is only visible to me”.

Consequently, we need to check for the possibility of universalization for this maxim as a universal law. Which is impossible to achieve without producing a contradiction:

The developer will not have learned to develop software in a vacuum, she will have read source code of her colleagues or of examples.

Additionally, a software developer is commonly not interested in “re-inventing the wheel”. That means that software products commonly depend on other, so called, *software libraries* to function. These are rather impractical to work with, if the developer does not have the source code at hand to check for invariants in the code or to work around lacking documentation.

But let us assume that the maxim only applies to the source code of big applications, not examples nor small snippets, as would be shared between colleagues. And let us also assume, that the external dependencies somehow work without being able to access their source code.

Even in this—quite favourable—situation would the application of the CI result in an immoral action because of the second possible contradiction. There can never be a desire to universalize the law, as it would contradict itself: The motivation behind the maxim is purely egoistical, but the universalization of the maxim would result in other people also taking advantage of it. Thus, the maxim would be impossible to universalize, as universalizing it would shatter the motivation to follow it in the first place.

Importantly, Proprietary Software is not only defined by access to the source code. Therefore, someone might actually allow full insight into the source code and still write Proprietary Software as the other two freedoms, not depending on the source code, could be violated. These are the right to run the program as you wish (freedom 0) and the right to distribute copies of the software (freedom 2). A maxim that could express the first case would be: “I want to control how a user runs my software”. Universalizing this *would* be possible: programs today even enforce certain limitations on the way they are run (trying to run a Linux-only program on Windows will obviously fail, as the program was not designed for this foreign environment).

But, this universalization attempt is impossible to justify whilst treating other rational beings as autonomous. Trying to enforce the way they have to run their programs, would entrench on their autonomy.

The mentioned limitation of programs not running in foreign environment notably does not entrench on the autonomy of other rational beings as it is never a feature but a lack thereof.

The second freedom must now also be analysed, as it could also be rejected. In this case the maxim at play would probably be: "I want to fully control the distribution of my software, and do not want copies of it to be shared". This is difficult to universalize without contradictions, as, like previously mentioned, software is often depended on other software projects. Thus, to distribute ones own software the dependencies must agree to being distributed. Which in turn means that the developer does not actually control the distribution of their software and that the maxim could only be universalized if software dependencies did not exist.

3 Conclusion

After having made the point that Kantian ethics in fact suggest, that the development of Proprietary Software is immoral, the fundamental problem still remains, that the whole line of argumentation relies on the theoretical concepts proposed by Kant.

These have time and time again been critiqued for not really being fit for the application on real world issues. The best known example, that comes to mind here, are the various contradictions found in the CIs themselves[13]:

> [A] stamp collector might live by the maxim, "I will buy but not sell stamps in order to expand my collection." If everyone were to follow this, then the collector wouldn't be able to buy because no one would be selling. This seems to lead to the implausible conclusion that collecting stamps (or collecting anything) is immoral.

Arguments like the one above can also be found for the others CIs. And although these can be alleviated by changing small parts in Kant's ethical framework—the quotation above would easily be defended by splitting the underlying maxim in two separate ones—holes can nearly always be found.

Possible alternatives to Proprietary Software include ideas like the "closed core" model, where only the core part of the software is Proprietary Software and the surrounding part is Free Software. These have not been evaluated, as these seem to rather uncommon today.

One other alternative to pure Free Software is the "open core" model, where the core software is developed openly, but then modified by the company behind it to add telemetry and other non-free software. One big example of this is the `Chromium` project, where `Chromium` is Free Software, but `Google Chrome` (Googles software product derived from `Chromium`) is not. These cases can not be evaluated directly, as the software being developed is without a doubt Free Software. But, the action that must really be evaluated is the act of forking (i.e. copying the source code) of the Free Software project, modifying it and turning it into a Proprietary Software project before releasing it. Which in turn would be exactly as immoral as writing Proprietary Software directly, as it does not differentiate itself from other Proprietary Software—the user is still not granted all four freedoms.

[13] Joseph Kranak: Introduction to Philosophy: Ethics, 2019, chap. 6, URL: https://press.rebus.community/intro-to-phil-ethics/, p. 57.

References

GNU-Project (ed.): Categories of Free and Nonfree Software, URL: https://www.gnu.org/philosophy/categories.html#non-freeSoftware (visited on 16/03/2024).

Idem (ed.): What is Free Software?, Version: 1.169, Feb. 2021, URL: https://www.gnu.org/philosophy/free-sw.html.en#make-changes (visited on 20/03/2024).

Idem (ed.): What is Free Software?, Version: 1.169, Feb. 2021, URL: https://www.gnu.org/philosophy/free-sw.html.en#four-freedoms (visited on 14/02/2024).

ISO/IEC/IEEE International Standard - Systems and software engineering–Vocabulary, in: ISO/IEC/IEEE 24765:2017(E), Aug. 2017, 3.2741, DOI: 10.1109/IEEESTD.2017.8016712.

Kant, Immanuel: Grundlegung zur Metaphysik der Sitten, AA IV, 1785, URL: http://kant.korpora.org/Band4/421.html.

Kranak, Joseph: Introduction to Philosophy: Ethics, 2019, chap. 6, URL: https://press.rebus.community/intro-to-phil-ethics/.

The HPG Logo in the header has been taken from this website: https://wpn.hpg-speyer.de/