2019

A Theory of Intellectual Property for Software

Stacey Kim
skim62@wellesley.edu

A Theory of Intellectual Property for Software

Stacey Kim

Submitted in Partial Fulfillment
of the
Prerequisite for Honors
in the Department of Philosophy
under the advisement of Professor Erich Hatala Matthes

APRIL 2019

ACKNOWLEDGEMENTS

Professor Erich Hatala Matthes,
Thank you for advising me throughout this long but rewarding journey. As with all intellectual works, this thesis required a considerable investment of time and labor to complete. I would not have reached the finish line without your tireless support and unfailing insight — they were the true incentivizing forces behind this entire effort.

Professors Corinne Gartner, **Ellen Hildreth**, and **Alison McIntyre**,
Thank you for sitting on my defense committee, challenging my ideas, and helping me grow. The knowledge I accumulated from your classes built the technical and philosophical foundations for this thesis.

Becca Pyeon, **Sophia Kim**, and **Nina Macaraig**,
My sisters in solidarity - your unmatchable wit, companionship, and love are invaluable to me.

And lastly, **Iyanuoluwa Adeniji, Tiffany Phan,** and **Jeanny Xu,**
Our decade's worth of friendship has been the greatest prize of my life. Thank you for always inspiring me to take on projects much bigger than myself. We may have left Victorville, but we never left each other.

TABLE OF CONTENTS

INTRODUCTION

Computer software expands into new frontiers at an exponential rate, pervading our daily lives. From the advent of the personal computer to the growth of artificial intelligence, software development opens new possibilities for economic efficiency, human flourishing, and creative expression in the digital age. With these new advancements, however, legislatures and courts face the increasingly difficult task of crafting intellectual property protection that is appropriate in scope and strength for the unique features and functions of software. They must additionally consider, on one hand, the social costs of granting exclusive author rights, and on the other, the need to incentivize innovation and ensure social well-being.

To strike an optimal balance between the two, I look to normative theories of property ownership. In Chapter 1, I first provide a brief overview of intellectual property law and survey the protections it provides. After establishing this legal framework, I introduce the Lockean theory of labor, which justifies property acquisition based on one's expenditure of labor on the shared commons. Because the labor theory, as Locke construed it, originally concerned fallow land and physical property, I challenge the theory's applicability to intangible works by showing that intellectual works are non-rivalrous in nature. The labor theory cannot justify intellectual property ownership because in granting exclusive ownership of inherently non-rivalrous goods, intellectual property regimes fence off the intellectual commons and artificially create a scarcity of resources.

The second normative theory of property that I consider is the incentive theory, which forms the backbone of many intellectual property defenses. The incentive theory argues that

property ownership incentivizes laborers to expend their energy and resources on inventing new products. Patents, copyrights, and trade secrets promote and sustain innovation because exclusive ownership rights allow creators to recoup their investment costs. The incentive theory therefore justifies intellectual property protection through the lens of a market economy. However, I show how the exploitation and monopolization of property rights disproportionately benefits large firms at the expense of consumers, thereby preventing the entrance of new market players and limiting consumer choice through vendor lock-in. Thus, despite the fact that the incentive theory posits innovation as its primary goal, it fails to improve social utility overall.

In Chapter 2, I detail the software rights granted by intellectual property law. Then, I argue that even if the labor and incentive arguments could justify intellectual property ownership generally, the specific case of software presents additional normative problems worth addressing. For one, I argue that unlike traditional material-based intellectual works, like novels and paintings, software blurs the categorical boundary between abstract ideas or functions and the expressions of those ideas. Software's unique qualities, such as its interoperability, dependence on copying, notational characteristics, and embodiment of the instructional process itself, push software outside the conventional paradigms of copyright and patent eligible works. I urge a reevaluation of software ownership protection based on the social need for software. I analogize the case of developing countries and the increasing globalization of intellectual property rights to the moral challenges that expansive software rights regimes present in an increasingly computerized society that values free information. In denying affordable access to intellectual assets that are critical to securing fundamental human rights, exclusive software ownership wrongfully perpetuates social disutility. Thus I argue that the intellectual property regime ought

to be more relaxed in its protection of owner interests, and even calibrated to fit the needs of the least well-off, when such protection is not only a barrier to innovation, but also a hindrance to social well-being.

Finally, in Chapter 3, I bring together these normative considerations to analyze a particular pathbreaking ideology that has risen in popularity within the software industry: the free and open source software movement. The free and open source movement is based on the philosophy that all users and developers alike should have free access to a publicly available intellectual commons. Specifically, the movement prioritizes source code availability in order to promote the social utility of software. After explaining the movement's underlying moral principles, I discuss the normative problems that the free and open source software movement encounters, as well as how the movement overcomes certain ethical worries.

I argue that the free and open source software movement adequately contributes to software innovation because software programmers may be intrinsically motivated, incentivized through recognition, or supported through alternate financing methods, such as offering support services and monetizing advertisements—therefore recognizing that software companies and developers ought to recoup their investment costs in the absence of exclusive intellectual property rights. Moreover, the free and open source software movement successfully captures the uniqueness of software because its licensing terms enable both the participation of diverse contributors and interoperable access to basic human rights, like education and public health. To conclude, I discuss certain caveats that the movement cannot perfectly capture in hopes that by acknowledging the flaws of various software ownership ideologies, we can effectively continue down the path of innovation and commodious living.

CHAPTER 1

NORMATIVE THEORIES OF PROPERTY RIGHTS

1.1 A Brief Overview of Intellectual Property Rights

Software is unique in that it is the only product subject to the four main domains of intellectual property law: copyright, patents, trade secrets, and trademarks. The breadth of computer software protection is attributable to software's complexity as a non-physical entity that is capable of both technical functionality and design variations. Copyright laws can protect software because its various design and concept implementations, like the program code or graphical user interface, classify as expressions of literary or artistic ideas. Simultaneously, the many useful and novel functions of software, as found in virus detection or word processing programs, fall within the scope of patent protection eligibility. Trade secrets and trademarks, too, grant ownership rights to software developers by protecting against the misappropriation of source code, brand names, and confidential information agreed upon in service agreements.

Although copyright, patents, trade secrets, and trademarks protect different elements of intellectual property, the protections that they grant derive from conceptually similar property rights. These property rights include the rights of access, use, revenue, modification, reproduction, distribution, exclusivity, alienability, and transfer. This section explores each of these rights in detail to establish a framework for analyzing property justifications as a whole. It provides a brief overview of copyright, patent, trade secret, and trademark law and the scope of their legal protections. By identifying the property rights at work within each legal definition, we can better understand how intellectual property functions through the framework of property

rights. Consequently, we can more appropriately contextualize software rights within our analysis of property ownership justifications in later sections.

The rights to access, use, earn revenue, modify, reproduce, distribute, exclude, alienate, and transfer all exist and operate in each type of intellectual property domain. In copyright, for example, the rights to exclude, distribute, and sell may be packaged together into licensing privileges, through which only the owner may distribute derivative versions of the software to the public (Freibrun). Meanwhile, software patent owners may hold the right to exclude (that is, prevent) competitors from duplicating their program algorithms, operating system methods, or other unique features. Competing parties are therefore excluded from distributing and profiting from such protected works. The concepts behind these property rights therefore not only manifest themselves in different ways among the four kinds of intellectual property protections, but also overlap amongst themselves within the same domain.

Take, for example, the right to modify, which allows owners to change features of their property. To modify one's property or good, one must first be able to access, use, and have control over it. The right to access therefore overlaps with the right to modify by serving as a precondition to it. This is evident in creative industries such as film, wherein a producer must first require the right to access a screenplay before acquiring the right to modify it. Similarly for physical property like farmland, an owner must have the right to access the property as a prerequisite to fertilizing, irrigating, harvesting, or using the land in some way. Other rights may also overlap with or mutually depend on each other. For example, the right to revenue entitles the owner with the sole right to profit from the property. It often co-exists with the right to exclude, in that all non-shareholders are excluded from generating profits associated with the protected

property. The right to exclusivity allows owners to exclude others from performing a particular action in regards to one's property. This can entail excluding access to a good, and prohibiting others from modifying the same product without the permission of the right-holder.

In fact, the right to exclusivity is ingrained in many of the other rights, such as the rights to reproduce, distribute, use, alienate, and transfer. The rights to reproduce and distribute grant owners the authority to circulate copies of their work and share derivative versions of it. For instance, a musician with intellectual property protection can permit or prohibit the duplication of a song performance. A novelist has rights over the publication and adaptations of their book, unless such rights are transferred or relinquished. Even for software, an owner has reproduction and distribution rights for a program, and can permit or prohibit others from making and sharing copies of the software.

The rights to alienability and transfer allow owners to freely dispose of their property, pass their property onto others, or convey it to another individual or party. Possessors of intangible property can surrender their control over their works, and re-assign them to any other person (Assignment and Cessation). For example, in the software industry, license agreements permit others to share in the owner's rights to access, use, and so forth. Businesses may also exercise the right to alienability and transfer through franchises, which are another form of licensing. While the franchisor exercises control over the company brand or any physical products associated with it, the franchisee shares in the right to revenue and the right to control, or micromanage, over different aspects of the product. These may include managing local marketing strategies and supervising on-site staff. By comparison, certain forms of property are inalienable and not subject to transfer on the part of the owner. These may include employee

security benefits, plane tickets, or the constitutional rights of life, liberty, and property. Thus, the four regimes of intellectual property may draw from and utilize the same set of rights, but apply them to different things that fall under the scope of their protection. Classifying these rights thus provides a framework for understanding the different protections granted by copyright, patent, trade secret, and trademark law. Equipped with this conceptual understanding of property rights, we can now define each type of intellectual property protection, and better understand their underlying rights.

Copyright

First is copyright. The scope of copyright protection is limited to the expression of ideas found in tangible mediums. Among the things eligible for copyright protection include, but are not limited to, literary works, artistic pieces, concerts and live performances, advertisements, and computer software. Copyright does not protect the idea itself. For example, an author of a physics textbook may have copyright protection over her explanation of a kinetic energy calculation, but she does not have control over the kinetic energy formula. Only the expression is protectable. Moreover, copyright protections do not extend to facts, ideas, methods of operations, and systems. While intellectual property law does not obligate creators to register a copyright, official registration allows the intellectual property creator to have a public record of their copyright claim and file an infringement lawsuit in court. With a registered copyright, the intellectual property author can have the right to exclude others from using, duplicating, profiting, and distributing the original work. The only exception to this is the fair use doctrine. Fair use allows a second party to use the original work without the author's permission for purposes that are personal or non-commercial, like commentary, education, research, parody, or

criticism. In this way, consumers may share some of the rights, such as the right to use, access, and distribute, with the intellectual property owners themselves.

Patents

Like copyrights, patents cannot protect ideas. Instead, they protect methods of operations, systems, and other inventions with functional purpose. Patentable inventions must be non-obvious, new, and useful (Freiburn). They can include inventions for new online systems, chemical processes, interfaces, mechanical designs, search engines, and more (Freibrun). Software products that feature data processing techniques, display presentations, program language translation methods, and other non-obvious, new, and useful features are therefore eligible for patent protection (Freibrun). On the other hand, patents cannot protect natural discoveries, such as animal species, and inventions that are too abstract, such as a mathematical formula not incorporated for any machine or process (Runge). Whereas copyrights cannot prevent others from creating similar expressions of a copyrighted work, patent rights explicitly prohibit others from exploiting or commercializing a work that is similar to a patented invention. Even if a second party independently develops a similar creation without any knowledge of the patented invention, patent law still prohibits them from marketing and selling their work (Douglas 487). The rights to exclude, earn revenue, use, access, distribute, transfer and the rest are thus equally applicable to the realm of patent protection. It is also important to note that patent law grants these property rights to the first party to file an invention, not the first person to develop a new invention (Intellectual Property Law). Understanding the difference between copyright and patent law will become important when we discuss how copyright and patent

distinctions are unsuited for the uniquely functional and expressive elements of computer software.

Trade Secrets

A trade secret refers to confidential information that businesses use for commercial benefit. Trade secret rights usually grant protection over confidential information by providing monetary remedies or injunctions in cases of trade secret misappropriation (Trade Secret). A holder of a trade secret therefore has the right to use the trade secret, profit from it, and exclude others from obtaining it. Trade secrets can range from the manufacturing process of a Toyota forklift to the formula behind Coca Cola. They encompass commercial methods of distribution or sales, advertising strategies, lists of suppliers and clients, and other compilations of information (Trade Secret). The only caveat to trade secret protection is that the holder of the confidential information must take reasonable measures to ensure its secrecy. A reasonable effort to maintain secrecy may look like establishing a confidentiality or non-disclosure agreement with employees, or it may involve implementing a computer security network. If an owner of a trade secret fails to maintain secrecy through reasonable measures, then the owner loses protection of the information as a trade secret (Trade Secret). In other words, under trade secret law, others may legally discover a trade secret through reverse engineering, or other means of independent discovery. These means of independent discovery cannot violate non-disclosure agreements or maliciously breach reasonable security measures (such as breaking into a secured vault) (Trade Secret). This is why many companies practice their right of transfer in licensing their products, rather than selling the entirety of their products without any restrictions (Trade Secret Protection). These licenses tend to contain “express restrictions against reverse engineering” and

define what users of the product can and cannot do in terms of their right to use, modify, and distribute (Trade Secret Protection).

Trademarks

The last form of intellectual property protection is trademark rights. Trademarks protect names, brands, and slogans of companies or products to distinguish them from the others (Trademark Basics). Although trademarks apply the same framework of property rights as copyright, patent, and trade secret protections do, I will focus less on trademarks in our overall discussion of software and intellectual property rights. This is because trademark protection is more concerned with securing an identifiable brand for a product or service than it is with any inherent idea or its expression related to the property itself. Nevertheless, it is worth noting that the various conceptual property rights reappear in every domain of intellectual property law, even for trademark protection.

Just as the legal system protects ownership rights over physical property, it aims to protect ownership over the products of intellectual labor. At its core, intellectual property ownership deals with the same rights and duties between owners and consumers. The most relevant of these rights include the right of access, use, revenue, modification, reproduction, distribution, exclusivity, alienability, and transfer. Copyrights, patents, and trade secrets encompass a combination of these rights. For software, the way these rights are shared between owners and users illuminate the philosophical stakes that arise when justifying either broad or limited intellectual property protection. With this in mind, we can now discuss the normative theories of property ownership.

1.2 The Labor Theory

The normative theories of property that we will cover implicitly assume a principle of self-ownership. Self-ownership is entrenched in natural law, or commonly accepted moral principles. It is the idea of having property in oneself. It grants exclusive entitlement and control to individuals over their own lives, bodies, talents, and thoughts. This sovereignty over one's physical and mental personhood restricts others from harming or interfering in the bodily rights accorded to each person. Although the ethical dimensions of self-ownership fall beyond the scope of our intellectual property discussion, its implications of promoting autonomy and well-being form the ideological backbone of each property ownership justification.

The Lockean theory of labor is no exception. In fact, among the various justificatory arguments for property ownership, the Lockean theory of labor, or simply the labor theory, has been the most influential in connecting the idea of self-ownership with the ownership of property. It represents the genesis of property rights and has been rigorously applied to defend or criticize intellectual property protections. The theory claims that laborers have a right to exclude others from the products of their labor, just as humans have a right to preclude other people from exploiting their personhood (Tuckness). According to the labor theory, actions of labor, such as the expenditure of time and energy on resources, entitle people to the products of their labor because by mixing one's labor with an object, one is also mixing one's own self (Tuckness). The natural right to one's body extends to one's actions and labor. The idea of self-ownership therefore encompasses ownership of the fruits of one's labor, too, because these fruits are an extension of oneself. In this sense, the Lockean theory of labor uses its central principle of

self-ownership to justify property rights over not only one's invested efforts, but also their resulting material products.

The argument is reconstructed and best understood as follows:

1. One has a natural right to, and property in, body and self.
2. Labor is an extension of body and self.
3. Because labor is an extension of body and self, one has a natural right to, and property in, one's labor. [From 1-2]
4. When a person labors upon, or mixes her labor with, an unowned resource, the resulting product embodies an extended part of oneself.
5. Therefore, the expenditure of labor confers and entitles property ownership rights over the resulting product. [From 3, 4]

The theory of labor also contains prerequisites known as the Lockean proviso. The Lockean proviso states that a person may mix her labor into common property or natural resources, and claim ownership over the resulting product only if there is "enough [of the resources], and as good [left in common] for others" after the acquisition of the property (Tuckness). The Lockean proviso implies that people cannot claim the property right of exclusion if the acquisition of the property renders others unable to use the common resources to create similar kinds of property for themselves (Himma 1153). A person who takes a common resource through her labor, and does not leave enough for others, wrongly commits misappropriation according to the labor theory. The proviso thus serves as a criterion for determining whether or not property privatization is justified. In summary, the proviso justifies a person's ownership of property only when it does not (1) waste the common resource and (2)

leaves enough of the resource for others to live well, given that available resources are limited (Tuckness). I will focus on the non-wastefulness and well-being elements of the Lockean proviso to show that even if the Lockean labor theory can be applied to justifications for private property generally, the proviso poses special challenges for granting similar rights to intellectual property specifically.

Non-Wastefulness

The first component of the Lockean proviso only allows ownership over a product as long as the product development process does not waste the common resource, and deprives others from their right to access and use it. In the case where a laborer appropriates more resources than necessary to create and own a product, the rights of others to those resources are wrongfully diminished either through wasteful neglect or spoilage (Moore and Himma). The non-wastefulness element of the proviso therefore invalidates a property claim in instances where the expenditure of labor on a common resource unnecessarily depletes the resource and limits its value for others.

Although Locke could not have anticipated modern-day frontiers in intellectual property, proprietary rights advocates still point to his labor argument when contending that "intellectual labor brings about the ownership of intellectual property" just as "physical labor brings about ownership of physical property" (Hick 110). For instance, the popular justification for intellectual property rights today emphasizes the voluntary efforts of content creators in imagining and developing intellectual products. These arguments defend an intellectual property owner's rights to exclude others and earn revenue from the products of their labor. Exerting energy in one creative endeavor and diverting attention away from other pursuits, after all,

involves opportunity costs that deserve an appropriate form of compensation. By permitting only property owners to profit from their creations, the rationale of intellectual property law implicitly embodies the labor argument's fourth premise: the fruits of one's labor are an extension of oneself. As such, they warrant and grant full ownership rights, such as the right to exclude, distribute, earn revenue, modify, use, and so forth.

The non-wastefulness element presents challenges to this justification of intellectual property ownership. Although it is true that most physical goods are depletable common resources, the proviso does not concern the domain of goods that make up intellectual property. The physical commons of tangible goods and the intellectual commons of intangible goods are not of the same nature. For intellectual property, the laborer builds upon the ideas of other intellectual laborers to create new works, rather than laboring upon some unused land or uncultivated crop (Chopra). The ideas that drive the creations of intellectual works are inexhaustible, unless copyright, patent, and trade secret protections preclude others from accessing the intellectual commons. For example, a work of Shakespeare may spawn modern day retellings, like the *Hamlet*-influenced classic, *Moby Dick*, or the musical adaptation of *Romeo and Juliet*, *West Side Story*, without excluding others from using the same Shakespearean source. Granting exclusive ownership of intellectual products as if they shared the same properties like tangible goods is to unfairly reduce public access to them and restrict their social utility (Chopra).

To better conceptualize how the labor theory and its non-wastefulness proviso is unsuitable for intellectual property, I will define both rivalrous goods and non-rivalrous goods, and clarify misconceptions about them relating to physical and non-physical products. A

rivalrous good is one whose use by one person prevents simultaneous use by another person. Rivalry therefore concerns the right to access and use. For tangible goods, one's possession of a physical object normally necessitates sole ownership because using a physical good generally deprives others of the same good (Child 579). For intangible products, however, the supply of resources is more often than not non-depletable. For instance, a fisherman's catch will prevent others from catching the same fish, but a performer's song cover does not prevent others from listening to the original song. In a similar vein, harvesting a farmer's crops for oneself makes it impossible for others to consume the food, but reading and re-reading an author's book does not prevent other consumers from simultaneously reading the creative work.

In the cases above, the tangible fish and crops exemplify rivalrous goods, while the intangible song and literature represent non-rivalrous goods. One might deduce from this that all physical goods are rivalrous and all intellectual goods are non-rivalrous. However, it is important to acknowledge that both tangible and intangible goods can be placed anywhere on the spectrum of non-rivalrous to rivalrous goods. Rivalrousness and its counterpart are simply features of a good that determine whether or not the good can be used without excluding others. Some physical property, like a forest scenery that visitors can enjoy, or a lighthouse that non-discriminately aids ships navigating the seas, may be placed on the non-rivalrous side of the spectrum. On the rivalrous side of the continuum, one may also find certain intellectual goods in addition to physical goods. For example, website domain names prevent owners from using a name already purchased by someone else. Whereas these types of rivalrous intellectual goods are scarce, the non-rivalrous intellectual property we are concerned about can be reused, copied, and

shared indefinitely without reducing the original supply. The non-rivalrous nature of intellectual property in such cases therefore defeats the proviso's worry of limited resources.

The consumption of non-rivalrous intellectual property therefore differs from that of most property subject to the Lockean proviso because the former is compatible with another person's use—and sometimes even simultaneous use—of the same idea, concept, expression, and so forth (Hick, 112). This further differentiates non-rivalrous intangible goods from rivalrous tangible ones because the shared use of intellectual goods can sometimes be required for the "full value" of some intangible property "to be achieved and appreciated" (Hick 112). For example, a language appreciates in value the more people learn it and communicate it with others. In contrast, sharing rivalrous physical goods, such as a car seatbelt or an umbrella, may not completely exclude everyone from the good itself, but it nonetheless would fail to maximize its full value. Unlike physical property, an intangible good's full value can still be achieved or appreciated through sharing ownership rights.

Intellectual property protection violates the non-wastefulness aspect of the proviso for another reason. When copyright, patent, and trade secrets grant laborers exclusive ownership of their intellectual property, the initially non-rivalrous intellectual property becomes rivalrous. Inventors may apply for patent protection over their products, and thereby restrict others from accessing the same methods, designs, or operations in the intellectual commons. If these inventors underutilize their intellectual products, then they prevent the most efficient use of their product. This practice is wasteful and directly opposes the non-wastefulness requirement. Granting exclusive ownership of knowledge goods violates the proviso because it takes products that are inherently non-rivalrous and renders them rivalrous through legal means. The labor

theory thus cannot justify intellectual property ownership because the intellectual property regime fails to meet the non-wastefulness requirement.

Well-Being

Nonetheless, it seems fair to argue that laborers should at least be adequately compensated when their labor is used by the public. This moral reasoning is evident in the U.S. Constitution's Takings Clause, which mandates that property owners are paid in return for any private property that is taken from them (Chopra). I concede the normative claim that laborers ought to be compensated for their invested work. However, I argue that there ought to be measures of compensation other than exclusive property rights, which prevents others from accessing the same resources necessary for their livelihood. My reasoning aligns with the second part of the Lockean proviso.

The second component of the Lockean proviso concerns the livelihood of others when a laborer claims a common resource for her own property. This well-being aspect of the proviso allows for private acquisition of goods as long as it does not deprive others of an enjoyable life, or worsen their livelihood. Like the non-wastefulness requirement, the proviso's well-being requirement must be met for the labor theory to justify property protection of both physical and intellectual goods. In the case of intellectual property, resources ranging from satellite communication to architectural design methods can easily occupy roles crucial to the well-being of society, whether for commercial, educational, or other social purposes. For example, the livelihood of patients in less developed countries would improve with access to innovative drugs and medical databases. I will reintroduce this concept in the second chapter to show how certain software that is useful to well-being ought not be subjected to intellectual property restrictions.

When the content of intellectual property is merely desired for non-crucial roles, such as entertainment or leisure, on the other hand, then the labor theory seems applicable to the extent that it does not violate the proviso's non-wastefulness element.

We must therefore question whether the rights of the laborer trump those of the user when it comes to each of their well-being. In instances where a rivalrous intellectual good is unnecessary for commodious living, the interests of the laborer in expending energy and resources, and thereby claiming control over her intangible property, carries more weight than the interests of other persons (Himma 1159). However, the interests of content creators might very well be outweighed by the vital purpose of the intellectual good in cases where intellectual content provides means for equitable living. This accords with the Lockean requirement that the laboring and use of a particular good must leave enough for others' enjoyment and survival. If intellectual content that is crucial for human livelihood is available at the discretion of a sole owner, then the needs of others may be unfulfilled. The fact that most intellectual goods are inherently non-rivalrous compounds this problem: intellectual property restrictions on the use of goods that could be used compatibly by all is not only wasteful, but also violates the well-being requirement of the proviso. Thus, when the sustainability or ease of human life is at stake, the collective needs of society should outweigh the individual right to have exclusive control over intellectual property, even if one has a natural right to one's labor. From this normative standpoint, one may concede that a laborer can still have the right to earn revenue, so long as this form of compensation does not take away others' right of access to the product itself.

The Lockean theory of labor therefore cannot be used to justify ownership of intellectual property in many cases because it will violate its own proviso. Not only is the supply of

resources abundant in the case of non-rivalrous intellectual property, but also the full value and non-wasteful use of intangible content may often necessitate compatible and shared use. Lastly, when juxtaposed with the needs of society, the individual right to one's labor pales in comparison. In such cases, the Lockean argument for labor fails to provide rational grounds for intellectual property law.

1.3 The Incentive Theory

Another argument for property ownership is the incentive theory. According to the incentive theory, the objective of intellectual property rights is to achieve an appropriate pace of innovation. The theory claims that failing to grant property rights, such as the rights to use, modify, distribute, and earn revenue, hinders innovation, whereas giving original inventors control and protection over their products ensures innovative progress. The incentive theory is rooted in the idea that property ownership generates returns on investments that an inventor would otherwise lose if a successor simply imitated the original product without incurring the same research and development costs. Investors and inventors alike would be discouraged from pursuing research and development expenditures if the cost of simply appropriating another's labor were significantly lower and more attractive than the expense of developing original products (Dam 338). Without continued research and development efforts, existing intellectual content and technologies would undergo little to no improvements over time. Economic competition would decline. New fields, such as the software industry, which are fertile with innovative potential face risks of stagnation as a result.

Proponents of private property rights therefore commonly appeal to the incentive theory in arguing that granting property rights best promotes the continued production and innovation of goods. The right to earn revenue and to exclude others from one's property incentivizes developers to compete against each other. Without some form of property protection to incentivize inventors and laborers, economic expansion, technological advancements, and artistic achievements would slowly come to a standstill. More importantly, the welfare of society that relies on such progress would suffer in the process. The discovery of new methods, product designs, and their evolving improvements are critical to fields such as medicine and agriculture. A poor or non-existent incentive system to ensure the continuation of such discoveries would severely harm the health and prosperity of society. The incentive theory thereby characterizes private property rights as the best mechanism to reward inventors, optimize the output and progress of intellectual products, and ensure social well-being.

We will first analyze the economic ideas underlying the theory to determine whether or not the incentive approach successfully justifies intellectual property ownership. In particular, I focus on how intellectual property rights drive competitive pressure amongst creators. I analyze how this competitive pressure may be essential to innovation and social well-being and what kinds of incentives the intellectual property regime generates. Second, I reconstruct the incentive argument and discuss how patent and trade secret systems cause worries for its premises. While incentives may be essential for the continued production and development of certain industries, like technology or the life sciences, overly broad or too narrow intellectual property protection can stifle progress in the form of patent monopolization or other means. Thus I reason that the incentive theory is flawed in its argument that intellectual property protection guarantees social

and innovative progress. In fact, I argue that it may not even be sufficient in achieving that end. The incentive theory does not necessarily spur innovation if the extent to which it is applied is overreaching. If strict intellectual property regimes limit the full social use of knowledge goods, then the economic efficiency in the innovation of intellectual works is gained at the expense of its distribution and accessibility. Lastly, I consider the possibility of incentivizing inventors through means other than intellectual property ownership.

The Market-Based Incentive and the Emotionally-Motivated Incentive

The incentive theory is largely based on economic reasoning. Even when applied to non-intellectual works, the incentive-based argument explains that property rights incentivize laborers to undertake intellectual labor. There are two layers of incentives worth distinguishing in this theory. The first and main incentive is generated by the market value of the property. The prospect of earning profit from developing desirable products incentivizes inventors to compete against each other, and spur innovation. Moreover, because laborers and inventors can recoup their research and development costs through intellectual property rights, they are better and financially more enabled to continue researching and developing against their competitors (Douglas 333). This resolves the appropriability problem, or the phenomenon in which innovators cannot reap financial gains from their products because the marginal cost of sharing and imitating intellectual works is zero. As I will later address, however, the appropriability problem may not be of concern to all innovators.

The second layer of incentives is an emotional one based on personal attachment to the fruits of one's labor. Although this is not true in every case, most inventors who first acquire a resource and expend labor on it value their product more than their hopeful appropriators. The

legal protection for the object of their attachment provides an additional incentivizing mechanism, without which one might be emotionally disinclined to contribute to the market. The increased value that these laborers place onto their product best leads to an efficient output of improved products, when that value is coupled with the assurance of financial protection. Granting rights to possess and profit from one's own ideas and labor therefore stimulates not only the production of intellectual goods, but also the improvement of such intellectual goods because inventors do not have to fear the theft of their property and the loss of efforts invested in those properties.

How Competitive Pressure Leads to Innovation

In addition to financially motivating investors and innovators, property rights serve as an economic force that fuels competitive pressure to produce the most desirable product available for social consumption. For successors, the competitive drive to succeed in the market leads them to innovate alternatives to the original product. Because these alternative products cannot infringe on the property rights of the original inventor, competitors must utilize their creativity to satisfy the market niche in innovative ways that are equally—if not more—desirable to society. In effect, the stream of products improves over time, as frontrunners and the original inventors must also continuously upgrade their products to maintain their market standing (Dam 356).

A clear example of this phenomenon is the MS-DOS operating system produced by Microsoft. Microsoft was the first to enter into the PC marketplace with its MS-DOS operating system (McCracken). Its successors could not infringe on the MS-DOS design because Microsoft had property protections over it (McCracken). Nonetheless, its successors delivered their own competing versions of an operating system, like Apple's graphical user interface-based operating

system (McCracken). Despite the increasing number of competitors, Microsoft was able to keep its market position by iterating new designs and improving upon MS-DOS with new software products, like Windows. By evolving to meet the demands of society (specifically, computing professionals and lay-users), Microsoft sustained its business even amongst its rivals, as well as influenced the modern computing landscape (McCracken). If Microsoft lacked intellectual property ownership over its operating system, however, competitors would most likely not have felt incentivized to expend their own resources on research and development costs. The competitors would have instead appropriated Microsoft's operating system, and sold copies of the leading product for a cheaper price. They would have prevented Microsoft from recovering the benefits of their labor investments, eventually driving the company out of business. Had Microsoft's competitors never designed the very alternatives to the original MS-DOS operating system, Microsoft may not have been incentivized to improve upon their product and release new product lines. Although consumers might have enjoyed spending less money to purchase replicas of Microsoft's original operating system, the consumers would have, in the long term, lost a socially optimal output of intellectual products. As this example shows, property protection spurs progress from an economic, and more indirectly, a social welfare standpoint.

It is important to note that, unlike the labor theory, the incentive-based argument justifies private property ownership because granting ownership rights is the best way to ensure a sufficient output of intellectual products to society (Moore 612). The incentive theory does not claim that content creators naturally deserve property protection or that mixing labor entitles them to sole ownership. Rather, the incentive theory focuses its attention on the needs of

consumers of intellectual works. Understanding this difference allows us to reconstruct the incentive-based argument as follows:

1. If a system of rights promotes societal well-being, then the social adoption and use of that system is justified.
2. Promoting societal well-being relies upon the continued development and dissemination of intellectual works.
3. The development and dissemination of intellectual works necessitate the incentives generated by a system of intellectual property rights.
4. Promoting societal well-being relies upon a system of intellectual property rights. [From 2 and 3]
5. Therefore, the social adoption of a system of intellectual property rights is justified. [From 1 and 4]

If all the premises of the incentive-based argument are true, then it follows that the conclusion must also be true. However, the third premise presents complications for the incentive-based justification for property rights. The incentive theory perpetuates the misconception that only material profits motivate the creative content-making process. Although intellectual property rights can be sufficient for economically incentivizing innovation, they are not always necessary for the original development, improvement, and dissemination of ideas. Some intellectual laborers, such as academics and artists, frequently produce creative works out of motivations other than financial gain (Himma 1153). Whether these motivations are based on egotism, moral duty, or something in between, authors may still create and share their works despite a lack of intellectual property protection. Hence, economic incentives may not be

necessary for creative development and innovation, like the incentive theory suggests, because the appropriability problem is not a principal concern for all authors. There are incentives other than financial motivation that society and its developers value, such as intrinsic motivation to contribute to general welfare. If the dissemination of new ideas and production of intellectual content can persist without the financial incentives generated by intellectual property rights, then the incentive-based argument fails to justify intellectual property ownership. I will explore these underlying alternative incentives next.

Despite the fact that the incentive theory rests on the notion that authors and artists will not produce socially valuable creations unless they are motivated by profits and other material guarantees, some authors may be incentivized to innovate through non-commercial motives. Some creators of intellectual works may find that proper recognition is enough for validating their invested labor and personal attachment to the product. Others may not even care about recognition as an incentive to contribute to general welfare. For instance, the vast majority of K-12 teachers are underpaid and underappreciated, yet many find the purpose of their work motivating and gratifying enough to continue contributing to children's education. The same kind of intrinsic motivation may apply to career public servants and other social workers who invest copious amounts of their time and energy into their work. Thus the economic incentive of intellectual property rights is not a convincing or necessary means of ensuring societal progress and the continued efforts of laborers.

Current intellectual property systems also contradict the idea that financial incentives alone are conducive to a healthy and competitive economy. The misuse of patent and trade secret laws exemplifies how intellectual property rights may not only be unnecessary to social welfare,

but also fail to even be sufficient for that end. In the case of patents, the strength of an intellectual property protection does not always correlate to an increase in incentive activity. While the patent system's initial purpose was to reward inventors and encourage intellectual advancements, the overly broad use and exploitation of patent protection sometimes lead to industry monopolization. For example, the amount of patent control that large firms possess has made it "almost impossible for new firms to enter the industry" (Hettinger 50). The competition-suppressing potential of patents often constitutes part of the reason why some private property rights proponents do not endorse every feature of intellectual property law (Himma 1152). Such features include a patent's duration of protection, in which others are excluded from developing similar inventions even if they do so independently or without knowledge of the original patent. This often makes it too difficult for others to compete in the market, and inhibits economic growth. Although a longer patent duration may incentivize inventors to a greater extent, it equally imposes "access and transaction costs disproportionate to the likely benefits from enhancing incentives" (Himma 1152). Therefore, it may not always be the case that incentives alone represent the optimal means to producing socially valuable intellectual goods.

Another worry relates to the overly broad application of patents. An overly broad patent covers the invention's functional abstraction rather than its functional structure. Abstraction-oriented patents limit infringement against a particular end goal, and cast a wider net than structural-oriented patents. Structural-oriented patents are more concerned with the invention's specific design or structure that allows it to achieve the particular end goal (Yang). The court case, *Abbvie v. Janseen* (Fed. Cir. 2014), is an example that clarifies the difference

between a patent's abstract and structural language. *Abbvie v. Janseen* claimed patent protection over a "sports car going 0 to 60 mph within X seconds instead of claiming a V-12 engine having certain technical features that enabled the car to reach 0-60 in so many seconds" (Yang). Patent claims like the one found in Abbvie v. Janseen are abstract-oriented. They contain broader language that is more limiting for competitors than the language of structural patents, which focuses on the particular invention's construction that performs the function. Because abstract patents do not have specific limitations like structural patents, they can exclude successive inventions from the market if they perform the same function. Broadly applied patents therefore lead to a higher likelihood that competitors will face infringement (Wiens). As a result, competitors will have to expend more resources and increase research and development costs more than is necessary, in order to create products that avoid patent litigation. A competitor may deem the payoff "insufficient for the amount of time and resources put into developing an invention" (Wiens). Thus, the mere existence of patents is not enough to induce innovation.

Trade secrets as well can stifle competition, rather than encourage it. If a company can sustain itself by exclusively relying on a secret advantage, then that company has no need to invent new strategies or constantly innovate to stay ahead of competitors (Hettinger 50). The trade secret system therefore represents an instance in which property rights may discourage further innovation or the development of new ideas. A recent case study of inevitable disclosure documents (IDD) show that stronger trade secrecy protections hinder innovation (Contigiani). The IDD authorizes companies "to prohibit a former employee from working for a competitor for a certain period of time, if they can show it would not be possible for the employee to perform her job without *inevitably disclosing* the company's trade secret" (Contigiani). The IDD

invariably makes it difficult for employees to leave the company. The company in effect disincentivizes disgruntled employees, who would rather switch firms, from maximizing their innovative productivity within the current workplace (Contigiani). Such strategies to protect intellectual property may therefore backfire and lead to a lower output of innovative products. Overly restrictive intellectual property regimes, like the IDD strategy, can have counterproductive implications even when employees eventually do move on to different firms. Because they are still bound by the IDD, they must produce inventions that are more "general-purpose" than innovative to avoid the risk of infringement or misappropriation claims (Contigiani,). Allowing employees to more easily switch between companies could act as a catalyst for increased competition, the continued production of intellectual works, and maximization of social utility. The third premise, which states that social prosperity and intellectual advancement both rely on the incentives stimulated by intellectual property systems, is therefore false. Intellectual property rights like trade secret protection may not even be sufficient for incentivizing innovation.

As discussed, the incentive theory incorrectly looks to economic incentives as not only the necessary condition for innovation, but also the solution to the appropriability problem. In the contemporary litigation setting, however, the appropriability problem is more complex than a mere competitive struggle between an original innovator and a follow-on successor, like the incentive theory assumes. There is an important distinction to be made between the types of competition involved. On the one hand, an innovating party that is first to a new market may face competition from a subsequent "innovator who tries to compete by putting a new and better product based on the original product into the same marketplace" (Dam 358). We will call this

type of follow-on innovation the "substantial improvement case" (Dam 358). On the other hand, a follow-on competitor may seek to profit from a competing product that closely resembles the original innovative product, but lacks the purpose of introducing a better product. We will call this type of case "me-too copying" (Dam 358). Whereas the substantial improvement case attempts to utilize the original invention as a "base or kernel on which to build enhanced or additional features," the me-too case only replicates the original product without incurring the initial research and development costs (Dam 358). The competitors in the latter case do not have the same intention of furthering innovation as do the competitors in the former case. Consequently, permitting me-too copying does not lead to innovative progress. It instead decreases the incentives for the original inventors to continue innovating. If the incentive argument promoted intellectual property rights because it was only concerned with me-too copying, then it could be less fallible.

However, the incentive theory faces criticism because it also presumes that the intellectual property system incentivizes more of the substantial improvement cases. The improvement on the original product may add value to users, and thereby contribute to innovation and social welfare. The incentive theory strives for this possibility by granting property rights to both the original innovator and follow-on competitor. Despite the added value of the second product, the reality of intellectual property law (mainly for overly broad patents) is that it views the follow-on product as a direct appropriation of the protected invention, and thus does not grant property rights to the latter. Even under copyright law, the use of the original invention as the follow-on product's base is still considered copyright infringement. This is true unless the follow-on inventor can adequately prove otherwise on the basis of transformative use,

or the use of an original work that alters its meaning with a new expression (Stim). The successor must therefore cease development of the follow-on product and extract it from the market.

What is needed then in place of strict private property rights are more, or equally, powerful incentives for stimulating production while preventing restricted use of socially valuable intellectual products. One alternative may be government funding and support of intellectual product development; another could be making the resulting product available to public ownership (Moore and Himma 8). Governments already fund research projects and characterize the resulting developments as public property, so considering this alternative may not be completely impractical (Hettinger 49). Unlike intellectual property laws which restrict the dissemination and use of intellectual products, government funding of intellectual labor would stimulate new inventions, support public access, and restrict monopoly control of valuable products (Hettinger 49). Although government funding is already limited, at least this alternative method of funding may avoid the problems of monopoly pricing (Moore and Himma 8).

Another alternative may be a price financing system, through which innovators compete amongst each other for a monetary award. Any private foundation, research institution, or governmental body can offer this kind of prize for successful research and development ventures (Baker 12). A famous historical example is the Royal Society for Arts and Technology's call for "a mechanical solution to replace chimney sweeps" (Baker 12). In rewarding a monetary prize, the price financing system mitigates the problems of intellectual property monopolization, and still makes room for follow-on innovation. This is because the party offering the prize would purchase the product "based on an assessment of its value," and make its social value publicly available through low licensing fees and distribution costs (Baker 11-12). Although this type of

incentivizing mechanism may have its own flaws, it would nonetheless provide the same kind of profit-earning incentives as an intellectual property regime, without excluding others from the social benefits of the product.

As discussed, copyright, patent, and trade secret protection may create incentives for product development, but nevertheless, the continuation and dissemination of innovative ideas do not depend solely on such material motivations. Moreover, such laws can in fact be antagonistic to innovation. If the purpose of private intellectual property systems is to promote social welfare and the use of information, then it is morally justifiable to modify these institutions or consider alternative models when they do not achieve this result (Hettinger 49). The possibility that intellectual property protection provides incentives for the production of innovative techniques and original works of authorship should not preclude any alternative mechanism that could produce similar, if not better, results. Even without the incentives promoted by intellectual property rights, a maximally efficient competitive enterprise may still be possible. Thus, the incentive-based argument for private property rights presents a shortcoming in its justification for intellectual property ownership.

The final takeaway is that innovation can occur and persist with or without intellectual property rights. The incentive-based argument derives its reasoning from the constitutional justification for intellectual property law: "to promote the progress of science and useful arts" (Hettinger 47). Absent intellectual property regulations, a competitor's best interest may be to seize and profit off the creative efforts of the original author without incurring the temporal and financial costs involved in producing intellectual works (Hettinger 48). But this does not automatically justify intellectual property systems as a necessary or sufficient means to ensuring

innovation. The misuse of patent and trade secret systems challenges the incentive theory's claim that property protection is a necessary incentive for content creators and intellectual laborers to contribute to the growth of intellectual products. Quite the contrary, several industries have witnessed innovation even without the protection of patents, copyrights, and exclusive intellectual property control generally (Boldrin 28). This is not surprising given that many of the contributions from innovators, intellectual laborers, and competitors derive from intrinsic motivations, most of which are likely to be more incentivizing than overly strict intellectual property regimes. The incentive theory looks to intellectual property systems as the only means of creating thoughtfully designed incentives, but there are other equally, if not potentially better, alternatives to encouraging innovation that do not undermine progress.

Conclusion

Thus far, we have discussed the limitations of the labor and incentive theory in justifying intellectual property regimes. Some aspects of their arguments are inapplicable to intellectual property, as opposed to tangible property, such as the conflict between the non-wastefulness proviso and the non-rivalrous nature of most knowledge goods. Other aspects highlighted legitimate moral concerns relevant to intellectual works, such as fair compensation for labor and incentives to motivate innovation, but approached them in a way that had detrimental and counterproductive effects. The heart of the following chapter focuses on how software ownership engages with these theories, given their particular worries. I argue that even if the incentive and labor arguments can be adopted for intellectual property generally, software poses special normative problems that the theories may not successfully address.

CHAPTER 2

PROBLEMS WITH SOFTWARE OWNERSHIP

2.1 Intellectual Property Rights as Applied to Software

Before we discuss problems with software ownership using the theories outlined above, I will provide an understanding of how intellectual property laws pertain to software. Computer software ownership concerns the same set of rights between owners and users as those of general property ownership. These rights are the rights to access, use, modify, duplicate, distribute, reuse, imitate, exclude, and alienate. An owner of computer software defines what particular rights are granted to the user through licensing agreements, which will become important to our analysis of the free and open source movement in Chapter 3. In this section, I will briefly explain how these rights limit or grant certain utilizations of computer software. I then clarify how copyright, patent, and trade secret protections affect computer software directly.

Software Rights

To the user, the most relevant rights are those that control what actions can be performed with her copy of the software. The right to access permits a user to read the software's source code (Douglas 486). The source code is the human-readable programming language that serves as instructions for the computer to translate into machine-readable object code and then execute. The right to use allows one to use the software for the intended purpose defined in the license agreement (Douglas 486). Meanwhile, if the license grants users the right to duplicate or reuse, then the user can permissibly "make identical copies of the software" or "recreate features or other aspects of the software in a new program, without using any of the original program's

source [code]," respectively (Douglas 486). The right to modify grants permission to change aspects of the software, and the right to distribute allows the user to give away any copies of the software. Lastly, the right to reuse allows one to utilize parts of the software, or even the entire software, in a new software program (Douglas 486). These are the rights that are most significant to the user. The rights to exclude and alienate are rights that pertain more to software owners, as they are the ones who can limit how and by whom their software is used.

Copyright, Patent, and Trade Secret Protections for Software

Knowing these intellectual property rights for software ownership is helpful for understanding how copyright, patent, and trade secret laws fit in the context of software. As noted in the first chapter, copyright protection extends to both literal and non-literal elements of an intellectual creation. The literal elements of software include its source and object code, whereas its non-literal elements can involve its file structures, data input formats, and other program organization sequences (Protecting Software).

Patent protection of software differs from copyright ownership in content and scope because patents concern inventions and grant a greater umbrella of protection. Software owners can seek patent protection of their product if it improves computer functionality in a non-obvious and new way, or offers a different solution for a computing challenge (Is Software Patentable). In terms of patent scope, software developers must be aware that they are still liable for patent infringement, even if they arrive at their creation independently, or without knowledge of existing similar software (Dam 368). They must consequently conduct patent searches to avoid such allegations. Unfortunately this is a costly and labor-intensive burden for software developers because large software programs contain hundreds, if not millions, of lines of code

that can implicate numerous patents (Goldman). At least for copyrights, independent creation serves as a sufficient defense to infringement claims (Dam 368).

One important similarity between patents and copyrights though is that neither of them allows exclusive ownership of an idea. Copyrights can grant ownership of an idea's expression, and patents can grant ownership of an invention that incorporates an idea into a useful function. Despite this aim, patents in particular tend to unintentionally extend protection to an invention's underlying ideas. This occurs when patents protect software methods and operations at too high a level of abstraction (Oz 164). For instance, a patent for virus protection software can broadly cover the ideas of filtering, rescanning, and matching code, if it does not explicitly describe an implementation process for these ideas. As we will see in the next section, the line between the expression of ideas and the ideas themselves is rarely easy to define for software because software has intrinsic functional and expressive properties. Not only does this produce an increasingly difficult problem for courts, but also slows down the development process for software creators. Any incorrect administration of patent rights for software is therefore detrimental to innovation because patent protection offers a greater scope of protection than copyright does. Thus it will form a significant topic of focus in this chapter.

Lastly, trade secret protection also extends to the domain of software. For example, trade secrets can protect the concepts in source code, computer algorithms, and other features of software. As long as software owners take reasonable measures in protecting the secrecy of such features, users do not have rights to access them. Now that we understand how these intellectual property protections relate to software, I will discuss how they cannot perfectly capture software's unique features and functions. Because of these unique features and functions, the

specific case of software presents an additional challenge to applying popular justifications of intellectual property.

2.2 The Uniqueness of Software

Even if the labor theory, incentive-based approach, and innovation arguments can justify intellectual property ownership, understanding computer software protection through the intellectual property framework is inadequate for capturing software's unique features and functions. Computer software legislation, intellectual property laws, and judicial decisions fail to understand the unique nature of software that distinguishes it from other protectable forms of literature and art (Durrell 232). Mainly, software's intrinsic functionality and use as a means to achieve an end differentiate software from the types of expressive works protected by copyrights. Software's ability to express ideas at a high level of abstraction simultaneously calls into question its eligibility for patent protection. Moreover, whereas traditional intellectual works discourage copying and often do not directly partake in the exchange of information, computer software embraces copying and interoperability as its essential functions. I therefore argue in this section that these unique features and functions warrant a reevaluation of software ownership justifications.

What is Computer Software?

To understand the unique and essential nature of software, we must first understand what computer software is. In the broadest sense, computer software is a collection of written instructions that direct a computer to perform a specific function (Durrell 235). A programmer writes these instructions in a human-readable programming language that is a combination of

pseudo-English and mathematical statements. These programmed commands are known as the source code, and will form an important chunk of our free and open source movement discussion in Chapter 3. Once the programmer completes the source code, the computer must compile them into a machine-readable, or machine-executable, language made up of binary digits. This arrangement of binary digits is called object code. After translating the source code into object code, the computer can then read and execute the code, performing the task at hand.

Altogether, the source and object code comprise software. Software is distinguishable from hardware because software, as its name might suggest, is intangible. Hardware, on the other hand, consists of the physical components of a computer, such as the hard drive and other tangible electronic devices. Thus while hardware neatly falls under the general category of physical property, software embodies the same intangible features subject to intellectual property protection. As we continue to examine the features of software, we will soon realize that software contains additional features and properties unlike those of traditional intellectual works. These elements may warrant a reevaluation of how intellectual property rights are applied to software as a whole.

Software as a Functional and Aesthetic Process

Among the various features of software, its source and object code deserve special attention because their written expressions can often misleadingly classify software as literary expressions. Although the syntactical construction of each line of code matters for algorithmic efficiency and parsability, the overall functions that the code carries out is independent of any particular organization of code. To illustrate, the source code of one software program can achieve the same functionality and result as another software program written by two different

programmers in contrasting grammatical form (IP and Software). This detail creates a fuzzy boundary between idea and expression because even though software source code exists in literal form, its non-literal elements are what dictate the functioning aspect of computers (IP and Software). As noted in the prior section, the opaqueness of this distinction complicates copyright and patent protections of software. Thus to overlook the non-literal features of software, and only perceive of a computer program as a literary work, would do an injustice to its unique relationship between the worlds of abstract ideas and tangible machinery.

Computer software is difficult to classify as a specific form of intellectual property subject matter due to other peculiar characteristics. Primarily, software occupies a functional role in that it instructs a computer, an aesthetic role in that it expresses the instructions, and a role beyond functionality and aesthetics in that it represents a process itself. For example, a person can improvise while following a recipe without re-writing the cooking instructions, but a programmer cannot change a software program's functions and results without mutating the source code directly (Watson). This characteristic of software makes it remarkably unique, compared to other intellectual works that may intend to either instruct consumers or produce artistic effects.

Comparing Software and Other Notational Systems

Other notational systems, such as furniture assembly manuals or orchestral scores, seem to occupy the same functional and aesthetic realm as software. Yet the functional dependency inherent in software is absent in other forms of notational systems. In other words, the functional purpose and output of a software program is strictly and specifically contingent on the input of software instructions themselves. All literal and nonliteral elements of a computer program are

made with the intended function execution in mind (Karjala). In contrast, notational systems that function as instructions, like an orchestral score, include other factors that generate their implemented results, like a concert hall's acoustics. The actual performance of the score depends on more than just the composition of the score. Musicians may take liberty in interpreting and performing the score, but nonetheless achieve the overall musical effect of the composition. Alternatively, they may even follow the notations correctly, and play every single note and phrase as written in the score. Yet, if their technique is unrefined, then the sonic result may not capture the intended emotional effect of the music. Software operates differently in that any change in the instruction itself will alter the resulting program (Watson). Thus, it is important to recognize that in the case of software we cannot easily differentiate between instruction and process, or expression and idea.

An instruction manual for constructing furniture, such as an armchair, offers a similar analogy. The instruction manual may be copyrightable as an expression of an artistic design or even protectable as a design patent, but the intellectual property protection does not extend to the physical manifestation of the constructed armchair. Copyrighting software on the other hand is akin to protecting both the instructions and the assembled armchair. Software is therefore unlike other notational systems that instruct a given process, and should be treated accordingly.

Conceptual Problems with Treating Software like Traditional Copyrightable Works

Despite the nuanced architecture and features of computer software, courts, legislators, and even the U.S. Copyright Office sometimes fail to recognize that computer software encompasses more than the simple code written by a programmer. In "Copyright Registration of Computer Programs," the U.S. Copyright Office declared that "[a] computer program and the

screen displays it generates are generally considered the same work, because most screen displays are created by the program code" (Circular 61). The U.S. Copyright Office thereby allows the claimant of program code and screen displays to file under the same copyright application (Circular 61). The conflation of the code and screen display assumes that the code is a fixed expression and that screen displays are merely the visual outputs of the expression (Compendium). After all, the screen display is part of a user interface, which determines the appearance of words and pictorial displays on a computer screen (Samuelson 564). However, this assumption fundamentally misunderstands that the user interfaces shares in the functional character of software code. Despite its visually expressive features, the user interface represents a "sequence of functions that occur when the user selects an option made available on the screen" (Samuelson 564). Without it, the user cannot interact with the program itself. By combining the source code and the interface display under one copyright, the U.S. Copyright Office misunderstands that there is more to software than just a written form of expression. Copyright protection is not supposed to extend to functionality, methods of operation, and the like, but in copyrighting the user interface, it mistakenly does just that.

Thus, even though it may seem logical to subject software to copyright protection, it is difficult to separate the written expression of software from its inherent functionality. It is true that physical limitations aside, the act of writing software code is analogous to authoring a literary text and other conventional art forms because the creativity, elegance, and usefulness of software is confined only by the imagination of the programmer (Durrell 234). However, the entire utility of a computer software program, its unique nexus between functionality and aesthetics, its embodiment of the process itself, and the totality of its inherent mechanisms make

software conceptually different from traditionally copyrightable, patentable, and trade secret eligible works.

The Nature of Copying

Another worry for software and its relationship with intellectual property protection is that copying forms an inherent part of the software development process. Of course, outright copying of source code is morally objectionable, but there is reason to consider the need to conform to, borrow, and build upon existing code for the purposes of social utility. Computer software developers possess only a relatively limited range of expression when it comes to configuring human-computer interaction. When software developers copy and conform to familiar elements of existing software programs, such as a standard user interface, the software becomes more accessible to the average user. This reduces the formidable learning curves that a user would otherwise have to face every time she attempts to install and use new software (Phillips). Moreover, allowing software developers to copy code prevents them from exerting extra energy and time into re-designing software products with a similar look and feel. This reduces potential barriers to innovation. Copying familiar elements of existing software therefore opens the market to not only users on the consumer end, but also new software developers on the player end.

Viewing software rights from a copyright framework is therefore counterintuitive to the essential copying nature of computer software. A computer program routinely performs copying functions, from duplicating files to making backup copies of data. As a legal system designed to prohibit copying, the copyright model is ill-suited to fit the function of software. Moreover, it overlooks the fact that independent software developers cannot maximize the social utility of

software if they cannot reuse existing code or make new implementations of existing programs. In effect, other software developers are inhibited in their ability to write new products. The type of copying necessary for software developers to continue improving software thus differs from the type of copying that underlies access to conventional, non-software forms of intellectual works. We should therefore be cautious in justifying copying restrictions for software because its essential function is so intertwined with the act of copying.

Software Interoperability

Software is also unique because of its ability to make applications and systems compatible with either a given hardware or another piece of software. This ability of software to communicate, share data, and exchange information between technological systems is called interoperability. Intellectual property regimes that exclude others from accessing, modifying, and reusing existing software therefore make it difficult for developers to realize the full potential of software's interoperable functions. Users also suffer in the process because the lack of compatibility and standardization across different devices decreases overall performance efficiency and innovation. Interoperability is therefore not only a unique feature of software, but also an integral part of many technologies and information systems necessary for daily use.

An example of an integral piece of software is the operating system. The operating system is a type of system software that provides a platform suitable for running other types of software, such as application software that runs specific user tasks. The operating system plays a major role in integrating program code, structure, and interface. It controls the intermediary communication between input and output devices, such as the keyboard, storage device, and display, and thus affects interface performance. Without the operating system, a user would not

be able to access important data and memory, or computer processes. An Apple computer's operating system, for example, illustrates the importance of an operating system. The operating system for Apple differs from that of an International Business Machines (IBM) PC (Durrell 236). In fact, the difference between each operating system is so stark that neither can exchange information with the other using the same software application. This incompatibility is the reason why software developers must dedicate more time, resources, and thought to designing different versions of software that are specific to each application (Durrell 236). The uniqueness of software's interoperable function thus separates software from the rest of traditionally protected intellectual works.

The inherent need for computer software interoperability also raises normative problems for labor and incentive justifications of software intellectual property protection. For a user to gain the most utility out of a software application, software must have a standardized interface capable of communicating with a variety of different applications and networks. A software's compatibility with a number of operating systems and "other software applications[] is necessary if the user is to deploy the computer as a gateway to communicate with other machines, telecommunications carriers, and software applications" (Bransco 152). Again, this notably distinguishes software from traditional forms of intellectual works protected by intellectual property laws. Museumgoers and readers may neither enjoy exhibitions of art that lack strikingly different techniques nor find copied and repetitive writing tasteful, but when it comes to software, users need uniformity and standardization of command structures, interface configurations, and other software environments in order to optimize productivity (Bransco 152). Again, the intrinsically functional nature of software sets it apart from other intellectual assets

whose main goal is one of artistic expression only. Because interoperability is necessary to promote computer efficiency, any legal attempts to inhibit literal copying of code, which is sometimes "necessary to achieve interoperability," is inefficient (Samuelson 1278). Software developers must be able to copy elements of code to standardize functionally optimal expressions. Requiring them to individually re-create such expressions in alternate ways not only adds to the labor of developers, but also leads to user incomprehension when these developers are discouraged from creating substitutes, and instead develop products that are incompatible with each other. Broadly based copyright and intellectual property protections therefore seem ill suited for software, whose interoperable features require deriving new works from existing code.

Conclusion

Because of software's unique features and essential abilities, we should be cautious of intellectual property regimes that attempt to categorically limit software as exclusively functional or expressive, like other intellectual works. The boundary that separates abstract idea and the expression of that idea is vaguely contoured and defined in the case of software. This ambiguity shows when courts allow copyright protection for software, even though software is intrinsically functional, and copyrights do not allow protection of ideas, concepts, systems, and methods of operations. Patent law also questionably protects computer software despite its abstraction of mathematical ideas. Additionally, software's unique interoperable capabilities present further challenges to the normative theories of property ownership. Even if the labor and incentive theories can justify intellectual property generally, they must address the special concerns raised by software. They must resolve the worry that restricting software copying and sharing requires more labor from independent developers, inhibits users from migrating to different platforms and

devices, and stalls innovation overall. We must recognize that software's unique functions and characteristics position software as a widely applicable tool, without which many social needs would go unfulfilled. Thus the uniqueness of software forms a critical dimension for understanding the social need for software, our next discussion point which outlines the particular consequences that arise from treating software as any other conventional form of intellectual property.

2.3 The Social Need for Software

Protecting computer software like any other conventional form of intellectual property overlooks software's extraordinary ability to promote human well-being. In nearly all sectors of the economy, one finds that computer software aids in increased productivity, and makes certain functions possible and more efficient. It assists in rapid information exchange in a global marketplace, quality control in national manufacturing, and effective communication in service institutions, such as education and transportation (Bransco 151). In a number of enterprises, software development is essential to social operation and safety, but there exist other areas where the absence of software would not significantly harm society. For instance, most gaming and entertainment software certainly supply access to the arts and may indirectly contribute to commodious living, but they are not as crucial for social well-being as software used for hospital records databases, national defense systems, air carrier communications, and other business applications. To account for these differences, I distinguish two classes of intellectual property to which software belongs: intellectual property with social need and intellectual property without social need.

Distinguishing the Categories

In the first class, intellectual property is critical to human flourishing and social well-being. The intellectual property in this category serve as a means to ensuring basic rights. I argue that patents normally protect this class of intellectual property because the patent system concerns inventions that are useful, novel, and non-obvious in nature. For example, Boeing recently patented a water harvesting system, which produces "water for industrial and human consumption in areas of the world where groundwater resources are scarce and rainwater collection schemes cannot provide adequate water supplies" (Brachmann). Even patents such as Walt Disney's System for "Robot based on human demonstration," which uses image capturing data to mimic human kinematic properties in robotic movements, has far-reaching social implications on the humanoid robotic industry (Robot). Escalator patent designs promote accessibility, solar panel inventions support environmental sustainability, artificial heart valves support bodily function, and cell phone utility patents allow for efficient communication. Software that also has clear and direct social value, like virus detection software and brain injury diagnostic systems, would fall under this category. Excluding access to these useful types of intellectual property therefore unfairly deprives people of a basic good needed for them to flourish and live well.

The second class, on the other hand, may be conducive to a pleasurable life, but is not directly necessary for human well-being. Access to artistic content, for instance, may enrich one's life with cultural knowledge and aesthetic pleasure, but cannot stand in for basic utilitarian needs. For instance, traditional forms of intellectual property, such as copyrightable novels and

artworks, may influence popular culture, provoke political discussion, and affect lives through generational influence, but they affect one's livelihood less frequently than intellectual property with direct utilitarian functions. Furthermore, the exclusive control over intellectual property in this second category does not deny others the basic resources necessary for commodious living. However, even within this category, technological advancements may build upon initially non-useful intellectual property and qualify it as useful software.

Other examples of intellectual property only marginally affect social well-being, if at all. For example, the patented design for bubble wrap has some social use but is not completely necessary for societal operation. Meanwhile, the "method for swinging on a swing" patent seems completely negligible (Method). Certain types of software, even with all its unique interoperable and copying functions, can fall under this second category, too. Amazon's One-Click, for example, allows consumers to purchase online goods in a single click, and Apple's slide-to-unlock feature provides a convenient method of activating a phone. Although the software capability most likely contributes to Amazon's increased sales and Apple's recognizable products, their businesses and the technological industry as a whole would not suffer major financial damages without it. Customers certainly could go on with their daily lives without such features. In a software context, user-friendly interfaces on web browsers, virtual environments, and editing applications promote higher efficiency, but the ease of use that these software applications facilitate contrast with the more direct means of social well-being that other software applications provide. Thus there are software products and processes that can be considered intellectual property with some social value, but not enough to deem them critical to human well-being.

That being said, it is important to acknowledge that the specific kinds of software that fall into either type of category is open to debate. While we are more concerned with the distinction between the two categories rather than the particular entities under each category, we can still plausibly conceive of how software might move from one category to another. For example, the healthcare industry has increasingly adopted virtual reality systems to distract patients from painful medical procedures. One such virtual reality system called SnowWorld immerses burn patients in wintery environments where they can throw snowballs, visit ice caves, and "perceptually escape into a pleasant alternative 3D virtual world" (Hoffman). Despite its entertainment origins, the visualization technology for SnowWorld reduces the pain stimulus of the burn patients and improves their overall well-being. Thus, software originally developed for gaming and other non-necessary pursuits can be modified and used for socially necessary purposes. Whatever software products happen to fall under the extensions of these two concepts, our primary concern is with this principal difference, particularly because software use is constantly evolving.

Socially Valuable Software and the Normative Theories

Given the widely applicable nature of software, I argue that the first category of software used for socially necessary purposes warrants distinct intellectual property treatment. I argue that neither the labor nor incentive theories for property rights can justify exclusive control over socially valuable software, even if we assume these theories can surmount the challenges identified in Chapter 1 and thus justify ownership of intellectual property that fall initially within the second class. I will show how strict intellectual property regimes lead to social disutility by

making it harder for software developers and users alike to realize the full benefits of socially valuable software.

Proponents of the incentive and labor theories emphasize the moral necessity of fair compensation and innovation. Even in the context of socially useful software, they may declare that denying a software developer ownership rights over her product wrongfully diminishes her labor and personal commitment to creating the software. It further discourages her and other developers from contributing to the development of similar software. In taking away a developer's right to exclusively own software, either by declaring the software public domain or lowering compulsory licensing rates, we wrongly exploit the intellectual labor of the software authors (Bransco 153). Arguments for software ownership based on incentive and labor theory claims therefore prioritize the interests of software authors, whose labor is necessary for software innovation, over those of consumers.

Despite the labor theory's concern for developers' interests, this defense of software ownership is inadequate because it fails to consider the utility loss of other developers. One developer's exclusive control over socially valuable software prevents others developers from improving upon the existing software, or from creating their own program using a similar software foundation. Strict intellectual property regimes build unnecessary barriers to software interoperability in particular because developers must pay the original author additional costs to access existing programs (Oz 163). If these additional costs are prohibitive, then they either force developers to "redesign every system to which they would like to make an addition or add a refinement," thus increasing their required amount of labor, or block the entrance of new developers completely (Bransco 153). In either case, innovation will stifle. Granting exclusive

intellectual property rights to the author of socially valuable software therefore contradicts the aims of the labor and incentive theories.

Advocates of the labor and incentive theories overlook not only the interests of other developers, but also those of users. By restraining other software programmers from accessing existing programs, intellectual property regimes prevent follow-on software innovators from debugging and improving their functionalities for users. This ultimately leaves the user with no choice but to wait longer periods of time to access improved software products (Bransco 153). Furthermore, a user may be unable to switch to another software supplier because of vendor lock-in. Vendor lock-in occurs when a consumer is dependent on the products of one particular vendor or supplier (Sjoerdstra). This not only erodes innovative progress, but also worsens customers' livelihoods because their use of the software is now limited to "the vendor's vision, requirements, dictates, prices, priorities and timetable" (Noyes). If legal protection of software excessively restricts competition and the further development of socially valuable software, then software protection laws significantly reduce the software's social utility too.

Weighing the Needs and Interests of Owners and Users

I suggest that we consider the strength of each interest when balancing the needs of owners and users of valuable software. Although an author is owed compensation to a certain degree, an author's interests may bear less significance or have less urgency than people's collective interest in intellectual content. When this content is as vital to human well-being as software in the first class is, we owe it to individuals and nations that they have sufficient access to the content (Himma 1159). Excluding them from the advantages that software offers, whether

it is to compete in a global economy or obtain scientific information, would not only fail to promote social efficiency, but also diminish human well-being.

In fact, technological advancements that depend on software in the healthcare industry could improve the conditions of poor communities to an extent so drastic that it would be immoral to restrict access to such software. Thus, although some measure of intellectual property protection is needed to promote efficiency, we must take caution of the fact that many existing intellectual property laws have inhibiting effects on society's overall well-being (Himma 1151). These inhibiting effects take detrimental turns when the intellectual property under protection is software that is instrumental to social utility and well-being. For these kinds of software, the value of the software to the user population is more important than the rights of the intellectual property holder (Hamister 16). Thus the livelihood of society that relies on software access is in moral tension with the right holder's expectation for fair compensation of software development. In the following section, I use an analogy of developing countries and their need for socially valuable intellectual goods to emphasize the moral importance of providing users the rights to access and use socially valuable software.

2.4 Intellectual Property Rights on an Global Scale

Even if we grant that intellectual property regimes promote economic efficiency and innovation, there are moral reasons to believe that private ownership of socially valuable software should be more relaxed, and even calibrated to fit the needs of the least well-off. In the public health sector, for example, restrictive access to drugs and medical knowledge imposes unfair disadvantages to those who cannot acquire treatment. In cases where human livelihood is

at stake, our moral intuitions seem to justify intellectual property piracy or misappropriation at the expense of author compensation. Furthermore, as we will discuss, cases of developing countries and their need for resources may show that a deliberate breach of intellectual property rights is morally justified. After all, the non-rivalrous nature of intellectual property enables its distribution to a larger population who benefit from the right of access without infringing on the author's own right of access. The misappropriation of goods necessary for societal well-being would violate the owner's right of revenue, but it would increase society's net well-being. On the other hand, the exclusion of the subset of intellectual goods that are vital to human well-being protects the owner's intellectual property rights, but results in net social disutility. The normative question is therefore to what extent intellectual property systems can justifiably protect and prioritize an owner's interests over those of the consumer population. I argue that the intellectual property regime should be more relaxed in its protection of owner interests when such protection is a hindrance to social welfare. I will examine a set of case studies in the emerging globalization of intellectual property trade and protection to prove this point.

In cases such as education, public health, and agriculture, there are reasonable grounds to value the consumer's well-being over the worries characterizing the incentive and labor theories. In fact, the reasons for justifying intellectual property systems—mainly, the idea that they motivate innovation—are insufficient to disregard the importance of advancing equitable standards of living. By analyzing how intellectual property regimes operate between developed countries and developing ones, I will prime the moral intuition that restricting the right to access and use socially necessary goods is unjust. The disparity of human welfare between developed and developing countries warrants revisiting the purpose of intellectual property rights. That is,

they should serve as a means towards greater social and economic welfare. By analogy, I will show that the subset of software necessary for human flourishing and well-being deserve a similar normative treatment under intellectual property law.

The justifications for intellectual property rights, as we discussed with respect to the labor and incentive theories, drive intellectual property policy on the international scale. Developed countries enforce strict intellectual property protections because they seek to recoup the costs of their labor and maximize profits. Although these protections restrict public access to socially valuable goods, developed countries may justify this consequence in the name of innovative progress. Incentives to produce intellectual goods will decrease without intellectual property regimes, and innovation overall will come to a halt. However, as detailed in our incentive theory discussion, stronger intellectual property protections do not necessarily lead to a greater output of innovative or socially useful products. Patent thickets, overly strict trade secret enforcements, and other intellectual property monopolies impede the dissemination of knowledge and industrial advancement (Bessen 1). Even within the domestic sphere, for example, broader and stronger patent protections over pharmaceuticals did not result in "an increase in the discovery of new chemical entities" despite intentions to incentivize innovation (Baker 6). The incentive theory legitimately concerns economic welfare, but the correlation between strict intellectual property laws and innovative prosperity is often misleading. When implemented globally between developing and developed countries, the intellectual property regime, as we shall see, encounters similar problems of exploiting intellectual property rights at the cost of social well-being, but on a more magnified scale.

The set of laws that establish intellectual property rights on a global trading system is known as the World Trade Organization's Agreement on Trade-Related Aspects of Intellectual Property Rights (TRIPS) (Intellectual Property). The TRIPS Agreement safeguards the interests of intellectual property owners around the world by legally recognizing their rights to access, use, profit, distribute, modify, reproduce, transfer, exclude, and alienate intellectual goods (Intellectual Property). Despite their efforts to promote innovation and economic welfare, the intellectual property provisions under these common international rules fail to satisfy the needs of developing countries. The Agreement disproportionately "serve[s] corporate interests in developed countries" for four main reasons (Baker 7). First, the monopolization of intellectual property rights prevents follow-on innovation, and therefore economic progress, in developing countries. Second, overly strict intellectual property regimes widen the gap of knowledge and resources required for developing countries to adapt foreign goods to their local conditions. Third, despite the incentive defense, the movement towards an international coalescence of intellectual property protection fails to innovate in areas most crucial to the welfare of developing countries. These areas include education, public health, and agriculture. Thus, even if a strong intellectual property regime serves as the sole catalyst for innovation, the direction of innovation fails to address the concerns of the least well-off—therefore, such a system cannot use the incentive theory as a moral defense. Lastly, we should seek to loosen international intellectual property protection for goods that are socially valuable because its consumers are more concerned with catch-up innovation rather than frontier or progressive innovation. In other words, their misappropriation or use of foreign products will not give them a competitive advantage over foreign laborers. Even though the labor theory claims that intellectual property

owners ought to be compensated, developing countries are not in a position to compensate foreign developers in the first place. The labor theory may still stand generally, but its application to intellectual goods that relate to basic rights is inappropriate because excluding such goods violates the non-wastefulness element of the proviso. The application of the labor theory to developing countries additionally violates the well-being component of the proviso because restricting the least well-off's access to valuable intellectual property worsens their livelihood. Thus, the inability to fairly compensate authors for the goods that the user has a basic right to shows that the restrictions of the good ought not exist in the first place. There are certain contexts where the inability to compensate is an inappropriate. Thus I argue that the incentive and labor theories cannot morally justify strict intellectual property regimes when they are applied in contexts where the intellectual good is necessary for one's basic rights.

Blocking Follow-On Innovation

As noted in the incentive theory discussion, the one major fallback of the intellectual property system is that it may impede innovation, rather than encourage it. The exploitation of intellectual property rights creates a barrier to innovation that affects not only domestic economies, but also those on international fronts. For developing countries, patent thickets harm economic development to a greater scale than that within a developed economy. Patent thickets are a portfolio of multitudes of overlapping patents that make it difficult for competitors to design around their protections of key technologies (Rubinstein 4). They are used to prevent developers from designing around a single patent, and they can prevent the entrance of new socially valuable innovations if such innovations are related to the original invention (Baker 65). In fact, in many scenarios, competitors may accept unfair licensing terms for useful inventions,

simply to avoid the risk of "costly and uncertain patent litigation" (Rubinstein 5). This is problematic because the original invention itself may not generate human flourishing, yet it may prohibit the production of follow-on innovations that are conducive to social well-being. When patent holders in developed countries monopolize their rights in this way, they block follow-on innovation that may be more tailored and necessary to the growth of developing countries (Baker 65). Follow-on innovation is necessary to improving the standards of living in developing countries because foreign products may not be specifically tailored to a developing country's needs. Developing countries require the right to use, modify, and access such original inventions to create more "varied products localised to domestic conditions" that the original developer may not have considered (Baker 7). The ability to adapt varied foreign products to a developing country's local needs is important because otherwise such inapplicable products will not have any social use or value. When the intellectual property rights of developed countries restrict the rights of their developing counterparts to modify such goods, they therefore create unfair legal barriers to societal well-being. If the rights holders of developed countries have too much protection of their ideas and broadly apply their right to exclude, then they undermine the capability of other inventors "to enter and compete in the global marketplace" (Reichman 7). Blocking entrance of new players to the market has the overall effect of slowing down innovation and increasing the risks of monopolization. For this reason, we should morally hesitate to extend and strengthen intellectual property rights for goods that are socially valuable.

Widening the Gap of Knowledge and Resources

Another reason why intellectual property regimes should be relaxed in the case of socially valuable knowledge goods is that they restrict the right to access the products of

innovation to those who need it the most (Baker 28). In effect, they widen the already existing knowledge and economic gap. When foreign countries enforce strong intellectual property protections, developing countries are restricted in their right to access, use, modify, and reproduce socially valuable goods. The high international standards for intellectual property rights limit developing countries' abilities to "reverse engineer unpatentable know-how, to add value by adapting foreign goods to local conditions" (Reichman 71). Preventing reverse engineering for socially useful intellectual goods prevents access to knowledge, and thereby prevents them from advancing technological progress and improving living standards. The education industry is a prime example of how strict intellectual regimes not only result in gaps of knowledge and resources between developing and developed countries, but also perpetuate them.

It is generally understood that the right to access education is necessary for human flourishing. The right to access education requires the ability to obtain and use educational resources, such as updated textbooks, teaching staff, and other sources of information. Limited access to learning resources severely impedes the social progress of not only developing countries, but developed ones as well. Stringent intellectual property regimes perpetuate this issue by prohibiting non-copyright or non-patent holders from reproducing, modifying (translating), and distributing the original work. Kenya, Malawi, Namibia and Zimbabwe, for example, have each raised concerns about the increasing number of students who did not own a textbook or had to share learning materials with peers (Baker 55). Cameroon faces a similar problem where only one reading textbook was available per dozen students (Baker 55). The social disutility apparent from this example is morally indefensible because the right to receive an education is a global human right (Universal Declaration). Even when textbooks are available

for free within the public school systems of advanced countries, the textbooks tend to be outdated and their quantities scarce, thereby contributing to the knowledge gap (Baker 55). If withholding textbooks from students is unjustifiable even with the understanding that distributing textbooks involves printing expenses, then withholding socially valuable software must also be unjustifiable, given that the uniquely copyable nature of software allows it to be distributed with zero marginal cost. Copyright protection of these textbooks may therefore satisfy the labor theory's concern for author compensation, but it does so at the expense of those whose well-being could benefit from educational resources. By deferring to owners' rights over users' rights, the intellectual property system unfairly creates prohibitive costs for learning materials, and in effect, exacerbates social inequity.

However, one may object that copyrights over textbooks are less of a legal impediment for consumers than patents are because copyrights concern the expression of an idea. Copyrights do not prevent governments or authors from developing countries from publishing their own textbooks to teach the same ideas (Baker 55). Although in theory this objection is sound, it overlooks the reality of the limited publishing and investment capacities of developing nations, who barely have established publishing industries (Baker 55). Additionally, it fails to recognize that developing countries are not in a position to finance or compensate foreign authors, whose intellectual works may be needed as a foundational basis for teaching and re-expressing the same ideas in the developing country. Developing countries ought not to incur formidable costs to import educational materials because these types of knowledge goods are necessary for their basic right to education. When calculated comparatively, the prices of books exported from the U.S. were exorbitantly prohibitive for students in Indonesia and Thailand (Baker 56). The study

found that a US $81.70 textbook on *The Pharmacological Basis of Therapeutics* cost Indonesian students the U.S. equivalent of nearly $900 "when compared using the GDP per capita calculated at purchasing power (PPP) exchange rate" (Baker 56). In Thailand, moreover, the importation and distribution costs account for almost half of the textbook price because publishers in developed countries restrict the right of foreign countries to reproduce, or reprint, books on their own land (Baker 56). When textbook costs for developing nations are this excessive, the labor theory can no longer justify the intellectual property system because the costs disproportionately overcompensate the author and simultaneously violate the proviso by restricting access to the intellectual commons. What may seem as normal copyright practices to developed countries is actually a form of copyright monopolization to developing countries, who are significantly disadvantaged due to the gap of knowledge, lack of resources, and restrictions of intellectual property rights. Thus for socially useful intellectual products that are necessary for basic rights, developing countries should not be expected to compensate intellectual property rights-holders. The fact that developing countries ought not to pay owners of socially valuable intellectual property in itself may suggest that intellectual property protection of such goods ought to be less strict, if not non-existent, altogether.

Even fair use exceptions are generally insufficient to morally excuse the harmful effects of rigid intellectual property regimes. The fair use doctrine limits the scope of textual access in both developing and developed countries. Although it allows access to a certain number of pages and a certain number of substantial ideas, developing countries must be able to access the entirety of textbooks and their ideas. Otherwise, the knowledge gap will continue to disadvantage students, and society as a whole, in the least-developed countries. Because

education arguably serves as the foundation for social progress, economic growth, and human flourishing, intellectual property rights regarding the right to access and use educational materials should be more relaxed. Copyright currently reinforces morally unjust barriers to educational access, and in doing so, makes it harder for the least well-off all around the world to close the knowledge and resource gap.

Innovating in the Wrong Areas

Even if intellectual property regimes spur innovation and ideally break down barriers to access, such a result does not matter from a moral perspective if the direction of innovation is not socially optimal. In other words, the incentive and labor theories cannot morally justify strong intellectual property systems if the protective measures do not innovate in areas conducive to social well-being, but only in areas that serve the interests of the already well-off. The latter situation is a global reality and a morally troubling one. On the international landscape, many innovators from advanced countries fail to address the most important needs of developing communities (Baker 28). In the drug industry, for example, the natural incentives of advanced markets are not enough to encourage pharmaceutical companies to research diseases most concentrated in developing countries. Despite their awareness of the risk to human life, companies in advanced countries are disincentivized to fund the research and development process to contribute to such human rights projects (Hayman). Thus, pharmaceutical companies may excel in their aim to innovate, yet disregard the areas of innovation that need it the most.

The problem of innovating in the wrong areas of social utility is not limited to the pharmaceutical industry. Even in agriculture, wealthy countries are disinterested in funding research where the soil type, local pests, weather, or climate of developing countries markedly

differ from that of developed ones (Baker 38). The incentives that patent rights give to creators of fertilizers and other agricultural products thus also promote innovation in a direction that ignores the needs of the developing world. For example, "seed companies may try to make farmers dependent on purchased seed instead of relying on replanting a portion of their own crop" (Baker 38). The patent system therefore shifts the focus of investment and labor efforts from the root of problems harming developing populations, to pseudo-solutions that are unsuitable for the least well-off but beneficial to those in power (Baker 38). Although such problems occur within advanced countries as well, "the risk for such abuses are much greater" in developing countries because of the "greater asymmetry in power and access to information in developing countries" (Baker 38). Extending or strengthening the intellectual property rights of pharmaceutical and agricultural companies, among other industries directly related to societal well-being, thus deserves moral reconsideration—particularly if doing so comes at the expense of the most vulnerable.

Competitive Fairness

The fact that developing countries are more concerned with follow-on innovation than frontier innovation morally challenges the labor theory defense in the international trading context, as well. When the incentive theory fails, developed countries often turn to the labor theory to justify stronger international intellectual property systems as a vehicle for competitive fairness. The labor theory ensures competitive fairness by creating a "level playing field" among intellectual property developers in the global marketplace (James 2). In the absence of a universally enforced intellectual property system, developers of intellectual products, "insofar as they seek market profit, will suffer unfair disadvantages in competing with those who are left

free to appropriate and capitalize on such intellectual goods" (James 2). Because those who misappropriate do not compensate the rights holders, the developers suffer from the violation of their rights to revenue. Advanced countries argue that developing countries who illegally reproduce and distribute their products in industries important to human flourishing "gain real but unfair competitive market advantages over developed-world innovators" (James 2). However, their idea of a fair level playing field is misconstrued. From an ethical perspective, the competitive advantages given to developing countries is morally justified because they begin at a significant disadvantage. Due to their disadvantaged starting points, developing countries would not reap a significant level of gain that would harm the trading capabilities of other companies. If we can morally justify breaching such protections, then it might call into question whether those protections should exit in the first place for goods that are socially valuable.

Other Consequences of Inflexible Intellectual Property Regimes

The TRIPS Agreement and its universal terms of high intellectual property protection unfairly burdens developing countries. By expecting developing countries to participate under the "same normative terms and conditions that govern advanced industrialized countries," strong international intellectual property regimes slow economic development (Reichman 72). Strong intellectual property norms even worsen the livelihood of international economies because developing countries must manage the expenses involved in enforcing foreign intellectual property rights, even if the countries themselves do not produce intellectual goods (Reichman 72). Thus, the consequences of stringent intellectual property protections affect more than just slow economic development. It often leaves the welfare of large populations in worse positions than they would be otherwise.

Interestingly, there are cases that prove the reverse effect, as well. The recent economic developments of Japan, Korea, and China exemplify instances in which minimal or weaker intellectual property regimes led to further technological and innovative progress within their developing countries. Rather than enforce strong intellectual property rights, these countries allowed for "creative imitation," and only loosely protected an original developer's right to use, access, reproduce, modify, exclude and distribute (Baker 30). The countries have exercised these freedoms to decrease the gap of technological expertise, and only afterwards, did they perform follow-on innovations to "promote a whole range of frontier technology industries," like cellular devices and Solar Cell Technology (Baker 30). The cases of Japan, Korea, and China thereby undermine the argument that strict intellectual property protection is necessary for technological development and innovation.

In summary, the monopolization of information in the industries of education, medicine, and agriculture highlights how the current international intellectual property framework raises the cost of social progress for the most vulnerable. They slow economic growth, shift innovation away from where they are most necessary, and widen the gap of knowledge by making resources expensive and inaccessible (James 3). Most egregiously, they limit "the policy flexibility that has been a hallmark of almost every development success story" (James 3). On the other hand, weakly enforced intellectual property rights have ultimately improved the economies of countries like Japan, Korea, and China. For all these collective reasons, we are justified in morally hesitating to extend intellectual property rights in cases where societal well-being is at stake.

Connecting to Socially Valuable Software

By analogy, when it comes to software, there are some cases where social disutility warrants a user's moral claim of software use over a developer's claim to prevent such use (Douglas 490). Just as strict intellectual property regimes inhibit the growth of developing countries, the expanding intellectual property rights over software cause worries for societal development. The monopolization of software rights creates prohibitive licensing costs and consequently prevents follow-on innovation, similar to the cases of developing economies. From a moral stance, the relaxation of intellectual property protections for socially useful software is not only justified, but also should be normalised. The most egregious problem that calls for this normative stance is the fact that the right to exclude others from accessing socially useful software worsens the livelihood of non-rights holders. Although there is a sense in which the exercise of any right to exclude might worsen the livelihood of non-rights holder, intellectual property that is fundamental to basic rights in particular deserves this normative distinction. The value of protecting rights of exclusion for this kind of intellectual property is insufficient to justify the negative effect on the non-rights holders. Excluding others from socially valuable software is no exception because this kind of software can facilitate access to the same fundamental human rights, such as the right to education. As I will show, the denial of access to software due to patent and copyright protections causes a digital divide, much like the knowledge gap between developing and developed countries. To prove my analogy, I will connect the moral circumstances that justify a less stringent international intellectual property system with those that affect software consumers.

Over-patenting Software and Aggravating Social Disutility

First, it is important to establish that restricting access to software use and distribution can lead to social disutility. When intellectual property regimes impose prohibitive licensing costs or harshly enforce the rights of developers, fewer people can acquire that software and use it (Douglas 487). By extension, there is a disparity between those who can afford software and those who are denied its benefits. Although the labor theory contends that only those who can afford to compensate the laborer have a right to access and use the fruits of that labor, the theory cannot be applied, at least justifiably, to knowledge goods pertinent to basic human rights. As discussed in previous sections, the uniqueness and essentialness of socially valuable software to human flourishing magnifies the benefits of its accessibility. The basic rights to education, food or agriculture, and safety (whether in the form of travel, environment, or something else) are arguably more enhanced and ensured by their integration and reliance on software. Correspondingly, the lack of access to the subset of socially valuable software worsens human livelihood to a greater extent than that of most other intellectual goods. This is most evident in instances of patenting trolling, evergreening, and other intellectual property right exploitations.

Just as the monopolization of intellectual property rights causes social disutility for developing countries, the phenomenon of patent trolling and evergreening software aggravates social disutility for even developed societies. Patent trolling is the process of applying for and owning patent rights with the intention to profit from bad faith litigation, infringement threats, or licensing demands (Gregerson). Patent trolls have no intention to develop the patented product or contribute to the pool of knowledge goods, and thereby hinder innovation (Gregerson). Evergreening is another egregious manipulation of the intellectual property system. Through the

evergreening process, rights holders seek to extend their patent rights and postpone their expiration dates by adding slight yet insignificant follow-on modifications to their inventions (Collier). These modifications rarely improve their products or the general well-being of their consumers. In fact, the evergreening strategy has resulted in socially malignant outcomes in the pharmaceutical industry because its use in monopolizing drug patents impedes competition (Collier). Thus, the application of either the incentive or labor theory is morally unjustified. In situations such as evergreening, strict intellectual property systems that enforce a developer's rights do not further innovation. Nor does it seem ethically compelling to compensate them for the prolonged period of protection that the evergreening process grants them. The issue is more pressing in the case of drugs because they are a products directly tied to public health, societal well-being, and the basic right to life.

These problems affect the software industry, too. Technology firms seeking to innovate new software products struggle avoiding patent infringements because patent protections are too wide in scope. The greater umbrella of protection that patents provide often crosses into fundamental items, like computer chips, and prohibits others from using such basic parts. This decreases the potential for follow-on innovation because the non-patent holders must obtain permission to use the protected invention. The fear of infringement litigation thereby leaves independent programmers and technology start-ups at a disadvantage because these types of developers cannot afford to spend more resources on discovering substitutes for the essential parts under protection. Patent trolling, blocking, and evergreening are just a few examples of the many mechanisms in which software owners can abuse the intellectual property system to the detriment of society as a whole.

At a closer look, we can see that prohibiting access to software through strict intellectual property regimes and rights exploitations worsens social disutility. It does so by creating unequal access to socially valuable software. Consumers who are financially disadvantaged cannot purchase the original software license, and enjoy the network benefits that software affords (Douglas 488). For example, because of pricing barriers, consumers may opt for substitute software that lack the same availability of program assistance, training, and file compatibility (Douglas 488). The problem affects the consumption of tangible goods, as well. Computer hardware, for instance, is a tangible, and by definition scarce, resource that requires software for its operation. By restricting access to software, intellectual property rights also strenuate the scarcity, utility, and distribution of computing technology as a whole (Douglas 492). This leads to an overall inefficient societal state if the disutility of excluding software access means resorting to poor performance technology and protocols.

A common objection is that in a fair market, a consumer must pay for a product to own it. Although this idea may generally stand as an economic justification, it should not hold as a moral justification for the subset of socially valuable software. This is because the type of software that is socially valuable facilitates an equitable standard of living when it is publicly available, and unnecessarily hinders well-being when privatized. For example, the unique power of software to organize, share, encrypt, and secure data is critical to the functioning of human rights organizations and efforts (Technology Tools). These organizations can more accurately and efficiently identify incidents of humans rights violations by performing online searches and database scans (Technology Tools). They can analyze these incidents, verify them, and circulate their findings with other organizations working towards similar causes affecting basic human

rights—all through software technology (Technology Tools). In such contexts, the value of human livelihood takes moral imperative and justifies a more relaxed ownership of software goods. Unequal access to software thus reinforces social disutility and creates a harmful digital divide, the focus of our next discussion. Thus the pertinent issue is not the kind of software at stake, but rather, the way software is used. Intellectual property rights ought not to apply to certain uses of software that, if made rivalrous, would reinforce social disutility.

Software Scarcity, the Digital Divide, and the Knowledge Gap

When intellectual property regimes tighten their rights to exclude, society faces prohibitive costs to access information. The knowledge gap that develops from such exclusion carries harmful implications and consequences, as we have seen in the case of developing countries. The similar deprivation of information applies to the subset of software that is socially valuable, as well. It is important to recognize that software licensing costs do not exclusively target and affect individual consumers. They also limit public access to software in libraries, schools, and other institutions that directly relate to the dissemination of knowledge (Douglas 488). This restrictive access affects students and patrons alike, who will be at a significant disadvantage compared to those who can afford to access the software (Douglas 488). The effect on social disutility does not stop there. Because software is arguably the most influential and effective tool for learning and advancing in our computerized society, limited access to software further disadvantages those who lack technical skills, which often aids in employment and career advancement (Douglas 488). Thus we ought to relax intellectual property protections of software so that others may affordably use substitutes of the existing software, access basic rights to education, and improve their welfare.

The digital divide is closely related to the knowledge gap affecting developing countries, as well. As an inherently non-rival good, software technology can promote human flourishing and standards of living in all countries without the worry of seizing any particular community of its consumer capabilities. To promote global well-being and growth, developed countries should relax their intellectual property strongholds, diffuse technological resources to developing countries, and remove impediments to knowledge transfer (Baker 29). The marginal cost of sharing software is so minimal after software's initial production stages that deliberately hoarding such useful technology, especially given the non-rivalrous nature of software, has the opposite and morally wrong effect of harming the least well-off.

Conclusion

It is therefore clear that neither developed nor developing countries benefit from stringent intellectual property regime practices. Overly strict intellectual property regimes lead to harmful consequences of patent thicketing, evergreening, and monopolizing knowledge goods even in developed countries, as we have seen in our software examples. The social consequences of this problem is exacerbated in developing countries, whose scarcity of knowledge goods perpetuates their economically disadvantaged states. In our era of information, access to knowledge goods is increasingly determinative of societal growth and flourishing. When intellectual property systems generate social disutility by stagnating the flow of knowledge goods vital to one's basic rights, we must reevaluate such regimes out of moral obligation.

Our moral intuitions thus call for a relaxation of restrictive intellectual property regimes, particularly for the subset of socially valuable goods, like certain software products. We can think of other means of fairly compensating software developers in a way that does not entrench

on the well-being of others. For example, monetizing software through advertising still financially compensates authors without allowing them to restrict others' rights of access. One person's use of a non-rivalrous good like software does not prevent another person's use of it, thus increasing net well-being and further justifying looser intellectual property regimes. The opposite scenario—in which the digital divide and knowledge gap widen because of stricter intellectual property systems—is an equal if not more motivating reason to relax such protections. In fact, in contexts where basic rights are contingent on access to such knowledge goods, we are ethically and justifiably inclined to deliberately breach such property rights. Our empirical and moral evidence in both the developing countries and software cases prove this point, and call into question whether or not the protections should exist in the first place. From education to the pharmaceutical industries, the incentive and labor theories fail to justify the regression and prevention of equitable standards of living. Thus we should keep all these normative intuitions in mind, as we analyze the free software movement's justifications and philosophies in the next chapter.

CHAPTER 3

THE FREE AND OPEN SOURCE SOFTWARE MOVEMENT

3.1 Introduction to the Free and Open Source Software Movement

Proprietary or private software ownership has come under scrutiny from different software developers and movements. Among them, the free software movement and the open source movement (which grew from the former) present the most interesting normative arguments. We will examine their ideological groundings and consider how they either successfully capture or fail to address the concerns of the labor, incentive, and social well-being theories. I consider the free and open source software ideologies under one collective umbrella, rather than separately, because their philosophical similarities outweigh their key differences. I ultimately argue that the free and open source software movement successfully avoids major objections raised in the labor, innovation, and social well-being arguments. Yet, there are other caveats within the movement's philosophy worth noting. In particular, the movement does not properly address the fact that certain categories of software are unnecessary for human flourishing. By overextending users' rights at the expense of developers' rights for these types of software, the free and open source software movement either overlooks or intentionally undermines the labor theory. The movement faces complications as to whether or not laborers should be compensated for the time, talent, and energy invested into products that are unnecessary for human flourishing. I thereby reason that while the free and open source software movement does not perfectly overcome every caveat, it best addresses the social need for software.

I will present my argument in the following way. In this first section, I will introduce background information on each movement's basic tenets, as well as their applications in the real world through licensing practices. While considering the different underlying values of each movement, I will emphasize how both movements are essentially concerned with the same set of licenses and categories of software. Thus, their differences are negligible in a broad sense, and we can hereafter refer to the movement as the free and open source software movement. The second portion of this chapter discusses the philosophy of the free and open source software movement within the purview of each theory of property. These theories are the labor, incentive, and the social need for software theories. By analyzing the movement against each normative framework, we can more effectively determine how the movement faces criticism not only from an intellectual property standpoint generally, but also in regards to software specifically. I argue that the free and open source movement adequately defends itself from criticisms about its lack of labor compensation, innovation, and committed programmers. Moreover, I point out that the free and open source movement ought to address preventative measures against exploiting alternate means of monetization to compensate for the lack of licensing costs. Despite these caveats that the movement should aim to address more explicitly, the free and open source movement overall points to a promising direction for the future of software innovation.

Background on the Free and Open Source Software Movement

Both the free and open source software movements center their philosophies around the accessibility of source code. Programmers should have the right to read, distribute, and modify the source code of all software, according to this philosophy (McGowan 411). Recall that source code is generally understood as a form of instructions that programmers write in a particular

programming language. This language is human-readable, and must be converted into object code, or machine-readable byte sequences, to instruct a computer's functions. Although this distinction between source and object code is not absolute (strictly speaking, all computer code is human-readable), software developers find that the source code is invaluable because it offers more convenient readability for humans (Touretzky). Computer programmers must have the right to access the source code if they hope to study, modify, and improve the software itself (Rosen 3). This is how the open source software movement acquired its name. When a software's source code is made publicly and freely available for all to access, it is considered open. By contrast, proprietary, commercial, or privately owned software usually guards its source code as a secret, in which case the source code is closed.

The free software movement shares in the open source philosophy, as well. It will become important for our labor theory discussion in the next section to know that the term "free software" does not refer to a monetary cost. Rather, the free software movement concerns a software user's liberties, such as the right to use, access, copy, distribute, modify, and improve software. The GNU Operating System, sponsored by the Free Software Movement, clarifies its name using the motto, '"free' as in 'free speech,' not as in 'free beer'" (What is free software?). I will explain these liberties in more detail when I discuss the licensing philosophies of both movements. For now, we must simply know that both the free software and the open source movements share the fundamental goal of software freedom, and both understand that open source code is the necessary means to achieve that goal. Thus, the ideological basis for both movements are the same.

The key difference is that the free software movement insists on the moral imperative of software freedom, whereas the open source movement focuses more on a pragmatic approach to integrating with proprietary business models. The two movements are ethically distinguishable by how they promote their philosophies and to what extent they are "willing[] to compromise[] these ideals in order to gain more acceptance [by proprietary, commercial software companies]" (McGowan 411). For example, advocates of the open source movement are willing to combine proprietary code in their open source software because this increases the chance of producing higher quality programs (McGowan 411). This approach also invites collaboration with firms in the software industry, who may want to commercialize proprietary derivative works from open source code. On the other hand, free software proponents believe strongly that proprietary code morally taints software programs. Thus, they insist that programmers should develop computer software using open source code only, and share their products freely and publicly with the user population (McGowan 412). In fact, free software advocates find that mixing open source code with proprietary code is so morally egregious that they object to the logic of the open source campaign. In "The advantages of free software," Richard Stallman, the architect of the Free Software Movement, states that considering the practical advantages of free software, as espoused by the open source campaign, is like considering the practical advantages of not being handcuffed—one does not need further reason to reject handcuffs (or proprietary software) because one knows what is at stake: one's freedom (Stallman).

Other than this ideological difference, both the free software movement and open source software movements are one and the same. Both movements allow flexible peer review and communal collaboration because of their open source code policy. Under each movement,

software consumers and developers alike can freely access and use the software, as well as change and add capabilities to the software, without any restrictions imposed upon by its original authors. Given these similarities, I compromise on the differences between both movements by grouping them together under the umbrella term of the free and open source software movement. By no means is this definition of the free and open source movement comprehensive, as it is still a point of contention among software developers and commentators today. However, the licensing requirements that I will outline satisfy the common properties found in both movements and will suffice for the purposes of the present discussion. Now that we understand the underlying values of the free and open source software movement, we can take a closer look at what software rights its licenses grant to both users and developers.

Software Licenses and their Inherent Philosophies

The free and open source software movement outlines several criteria in its licenses which differentiate it from other software ownership philosophies. Because license agreements confer specific software rights to both users and developers, I will describe the main idea of each licensing criteria important to our understanding of the free and open source movement ideology. In the next section of this chapter, I will examine each idea more closely within the framework of each normative theory of property. This way, we can conduct an ethical analysis of the movement in an informed way.

Licenses are legally binding conditions on the use of intellectual works. We can think of software licenses in particular as contracts between developers and users that convey limited rights of software use to the recipient of the license. For example, a license may allow the purchaser of the software to access, read, and use the program, but may prohibit the user from

modifying or redistributing it. Licenses do not directly affect the code but, depending on the license, can certainly affect what a user or another programmer can do with it (Watson). We are interested in licenses within the context of the free and open source software movement because licenses embody the particular philosophies of their software vendors (McGowan 417). By understanding what conditions the free and open source movement enforces under intellectual property law, we can better understand its inherent normative stances on software ownership in general.

Licenses that are acceptable under the free and open source software umbrella must comply with the following criteria:

1. Free Redistribution

Licensors are not obligated to distribute their software's source code to everyone, but they must make it free of charge upon request (Rosen). Another way to think about this is that licensees do not have to pay royalty fees to a licensor in order to distribute or sell additional copies of open source software (Rosen). This promotes higher quality of code and software innovation overall because it "eliminate[s] the temptation for licensors to throw away many long-term gains to make short-term gains" (The Open Source Definition). When we apply the free and open source software ideals against a normative framework in the next section, I will examine whether or not this criterion fully meets the needs of the incentive theory.

2. Source Code Availability

As mentioned earlier, a program must include source code and make its source code publicly available to be considered free and open source software. The software must publicize its source code in an accessible, "un-obfuscated" way, so that future developers can add

modifications (The Open Source Definition). By facilitating modifications, the source code availability criteria encourages software evolution and makes innovative progress "a practical reality" (Rosen).

3. Modifications and Derived Works

The license must grant the right to modify and copy the original software (The Open Source Definition). Derived versions of the software must include the licensing terms and conditions of the original software (The Open Source Definition). The ability to redistribute, study, and experiment with modifications enables faster innovation, according to this clause.

4. Integrity of the Author's Source Code

With the right to modify, derive, and copy, software users have a right to access information about the original author of the software. Likewise, software developers and maintainers have a right to "protect their reputations" (The Open Source Definition). What this means is that in addition to making all modified versions of the source code available, free and open source licenses may permit distributions of the original source code along with patch files, or added software updates (The Open Source Definition). Licenses thus often require derived works to have version numbers that are different from their original sources (The Open Source Definition). As a result, everyone can track the changes to the software and distinguish them from the original software source code (The Open Source Definition). By marking modified versions of the code, programmers can prevent the likelihood that users will mistakenly attribute program errors or improvements to authors of previous versions (What is free software?). I will address this point in more detail when we analyze the free and open software movement using the labor theory.

5. Distribution of License

There is no need to create an additional license when redistributing the software program to others. This is because the program's license also applies to all recipients of the redistributed program. The rationale for this clause is to prevent exclusive access to software through "indirect means such as requiring a non-disclosure agreement" (The Open Source Definition). This avoids the problem of trade secret exploitation and intellectual property monopolization, as discussed in earlier chapters.

Summary and Anticipated Problems

The licensing provisions outlined above are a result of a compromise between the ideologies of the free and open source software movements. The open source movement advocates for further provisions beyond those outlined above, such as freedoms to combine open source software with any type of proprietary technology and other software. However, these sorts of clauses violate the free software philosophy. I excluded these clauses from the overall license guidelines because the collective free and open source movement does not consider such externalities as part of its shared ideology. As we continue to the next section, I will reveal how the exclusion of these clauses may pose ethical problems for the free and open source movement.

3.2 Normative Analysis of the Movement

The Ethical Tension Between User and Developer

This section will conduct an ethical analysis of the free and open source software movement. We will apply the tools from our normative framework to evaluate how the movement either disregards or successfully overcomes the worries of the labor, social

well-being, and incentive theories. As we have seen in its licensing terms, the movement is conceptually grounded in the relationship between users' rights and developers' rights. We will investigate how the tension between user and developer interacts with the various normative theories. If the free and open source software ideology can avoid the objections raised by various normative concerns, then the movement itself may point to a promising, ethical direction for the future of software development. I argue that although the free and open source principles successfully address the concerns of the incentive and social well-being cases, the movement fails to entirely capture the ethical concerns of labor compensation in certain contexts.

The Incentive Theory

The free and open source software movement argues that quality software is dependent on access to earlier versions of software and their source code. By permitting modification and redistribution of derivative works, the free and open source philosophy encourages unrestricted creation. The first, second, and third provisions of the movement's licence agreement allow for exactly this. Licensors cannot impose monetary restrictions, such as royalties, on software if someone wants to use the original software. The free and open source software ideology claims that technological progress is a byproduct of this policy. If source code is the medium that software developers need to create a computer program, then removing barriers to all source code will increase the likelihood of collaboration, dissemination, and innovation. We will contextualize this claim within the incentive theory to see if it can support its claim against our objections.

The first objection is a familiar one. Without royalty payments or financial earnings for their labor, software developers will invest less effort into creating innovative products

(McGowan 421). The argument for the incentive theory, as we recall, states that intellectual property protections spur innovation because authors can generate profit from their creative works.When free and open source software licenses not only prevent owners from profiting off the distribution of their works but also depend on the upheaval of private software ownership rights, then software creators no longer feel motivated to develop meaningful programs. The advancement of technology and innovation directly relates to whether or not intellectual property creators "believe that the system rigorously protects their creations" (McGowan 410). The uncertainty that one's labor may not generate income will disincentivize potential innovators. Software innovation as a whole consequently erodes over time.

Moreover, even if software developers were incentivized to contribute to the free and open source initiative, there is no guarantee that a given software product will garner the support of developers for a sustained period of time. In other words, the free and open source software license fails to provide rights to developers that incentivize the investment of their talent and labor long-term. The longevity of the development process for a given piece of software is indeterminable. The development process may cease to persist over time because the software programming community has no obligation to commit to and update new versions of a particular product. Furthermore, software developers may be intrinsically motivated to contribute to free and open source projects, but this sort of motivation may diminish when faced with the painstaking task of debugging and testing code. If programming issues for socially valuable software are left unaddressed, then innovation overall significantly decreases altogether.

This contrasts with the proprietary software environment, where commercially-backed programmers have a concrete, financial incentive to commit to a software project through all its

stages of development. For example, proprietary software often undergoes routine updates because vendors can increase sales from paid upgrades. The incentives of proprietary software licenses therefore ensure continued innovation, unlike those of free and open source ones. This calls into question whether or not the conceptual distinction between "free speech" and "free beer" is applicable in the free and open source practice. Given the examples mentioned, it may be unlikely, and perhaps even impossible, in practice for one to make software free in the free speech understanding without also making the software free of charge. If the two are in fact inseparable, then the free and open source movement must address how it will overcome the innovation and labor compensation worries.

Software support services may provide a counterargument to this objection. Although software companies under the free and open source software license must make their software code freely available, they can still generate profits from offering support services for their free and open source products. Software developers may also earn revenue from advertising partnerships or add-on subscription sales (Rubinstein). These profits can serve as financial incentive to continue investing in free and open source software development. An example of this business model is the software company Red Hat. Red Hat earns enough profit from offering technical support services to its open source operating system, Linux (Rubinstein). Even though its licensing terms restrain Red Hat's right to profit from Linux directly, the services necessary to support Linux nonetheless garner enough value from customers to earn large financial returns. Because support is where free and open source software developers earn their revenue, the program itself usually includes comprehensive documentation, updated online forums, and even live customer support chat (Noyes). This cultivates innovation and collaboration within a

community of software developers to a far greater extent than what proprietary software vendors provide. Moreover, software developers, companies, and start-ups can monetize freely available software through advertising, in addition to customer service support. By doing so, the free and open source software movement can uphold its "free software as in free speech" licensing philosophy while still granting its authors some form of financial compensation.

The free and open source philosophy also overcomes the objection that free and open source licenses lead to an unsustainable reliance on volunteer programmers, who are less committed to improving software long-term than their paid counterparts. There are two reasons why this objection fails to undermine the movement. The first counter-argument focuses on the equally flawed system of proprietary software development. A proponent of the free and open source philosophy can claim that the financial incentive granted by exclusive software ownership rights does not automatically lead to higher quality software products. For example, it is common for proprietary software developers and vendors to abandon their projects when they no longer generate return on investments. On the other hand, free and open source software developers can produce higher quality software because they do not have the same financial incentive to rush their product to a market and earn profit (McGowan 420). Moreover, proprietary software developers sometimes purposefully leave defects in the software to motivate consumer anticipation of future modified versions (McGowan 420). Thus the incentive to innovate is in the wrong direction, similar to the tendencies of developed countries to produce goods that are profitable domestically but ignore the needs of developing countries. However, with free and open source code, more programmers can access software code to test for any bugs. This contrasts to the proprietary practice of excluding all outside entities from viewing and inspecting

the code. When vendors limit who can check for vulnerabilities in their software code, they increase the likelihood that flawed code will go unexposed, and decrease the rate of innovation. On the other hand, a more relaxed approach like the free and open source license not only increases the likelihood of discovering bugs—and thereby spurs innovation—but also proliferates trust within the community of volunteer programmers and users (Noyes).

The second counterargument re-emphasizes the moral incentive of contributing to the greater good. Programmers who hope to prevent proprietary competitors from monopolizing software products and from gaining unfair advantages over individual programmers may gravitate towards the free and open source software model regardless of financial incentive. Restrictive patenting of programming techniques may frustrate individual programmers, who understand that intellectual property innovation relies on the access to and improvement of existing knowledge and earlier versions of software, respectively. Even if the free and open source software licensing terms narrow the avenues for financial return, the sustainability of the movement offers greater incentive for continued participation. Without the free right to access open source software and the requirement to share such developments with others, individual programmers would be unable to use and improve upon fundamental programming techniques, software innovations, and discoveries in the first place. This moral crusade to perpetuate source code availability may in itself motivate software developers to continue their contributions to software projects. In essence, the free and open source software movement acknowledges the fact that software development is a social product not only in its development process, but also in its application. Without access to the source code and the ability to peer review and modify it, fewer people will benefit from the software.

Societal Well-Being

The social benefits of software development is another important point of focus within our normative framework. According to the societal well-being concern, intellectual property protection should promote human flourishing, and avoid impeding one's basic rights. The free and open source software movement addresses the normative concern of whether or not software authors can deny users various rights of ownership (Douglas 486). The movement centers its philosophy on the idea that software licensors cannot and should not withhold the right to modify and redistribute software from users, who have a civil liberty to their benefits. Given this argument, we will further probe free and open source principles to see if it withstands objections from our societal well-being theory. From there, we can more confidently evaluate the robustness of the free and open source software movement from an ethical perspective.

The common objection to societal well-being is that the author has a morally vested interest in the product as an extension of self and labor. As we argued in the social need for software and the international context of intellectual property trade, however, there is a compelling ethical reason to value the interests of users more than that of intellectual property owners. This is a plausible belief, considering that some categories of intellectual property and software are critical for the survival and flourishing of humanity. Thus, according to the societal need for software argument, the needs of the greater user population outweigh whatever moral right the software author has in the original creation. However, there is another objection worth considering in the context of free and open source software.

One can argue that proprietary or private software ownership is morally justified because it does not make other developers and users worse off than they would have been without it.

After all, the public did not have prior rights of access and use to the software and its benefits. It is not until the author produced the software that her exclusive control over the creation seemed to deprive others from it. If this is true, then the free and open source software philosophy is based on an illogical premise. In reply, advocates of the free and open source movement might point to a similar analogy of developing countries' needs for access to scientific research and medical knowledge. They can claim that proprietary licensing or exclusive control over socially valuable intellectual property in fact does worsen the livelihood of others. Consider the following thought experiment, inspired by Himma's example of cancer patients: within a particular region, a viral outbreak threatens the livelihood of population *X* (1156). Either by chance or through years of toil, a researcher discovers the singular cure to the epidemic. Yet, the researcher withholds the cure from those affected, and claims sole ownership of her new discovery. Although the population did not have access to the cure before the researcher discovered it, the population is now worse off because their chances of surviving are now reduced by the researcher's exclusive appropriation of the cure. The same thing can be said of worsening the plight of farmers (and, indirectly, consumers) by withholding new agricultural knowledge during drought seasons or famines.

Even in the computer realm, we can think of strong ethical reasons why certain software products not only promote human flourishing when openly shared among users, but also worsen people's livelihood when they are exclusively controlled. For example, voting software is integral to democratic processes (Wolf 21).The free and open source license permits public examination of software source code, which enables a more transparent and participatory election (Wolf 21). Free and open source principles successfully capture the idea that the

freedom to access and inspect the voting source code is more likely to guarantee fairness in the electoral process (Wolf 21). Of course, this is not to say that the free and open source licensing permissions to freely modify and redistribute voting software is optimal for a constitutionally defined process. Rather, I claim that the mere ability to freely access the voting source code would certainly cultivate more trust and well-being within a citizenry than would any strict and closed proprietary control. Therefore, the free and open source software movement holds its ground when faced with the objection that society has no moral interest in the property of others.

Although we touched upon the labor theory in this discussion of social well-being, the theory is still worth considering independently because it raises a normative complication for the free and open source software movement.

The Labor Theory

The license provisions under the free and open source software philosophy preserve the rights of users and the community. At the same time, they require software developers to relinquish some of their own rights. Developers' rights to distribute, profit, and exclude are severely limited in contrast to the rights afforded by proprietary licenses. Although the free and open source software license seems to oppose the labor theory, its values directly parallel the ideology of the Lockean non-wastefulness and well-being proviso. Even so, concerns about proper labor compensation may be enough to reformulate and strengthen objections raised by the labor argument. In this section, I examine how the movement fares in addressing these particular worries.

According to the Lockean proviso, appropriation of a resource is justified if there are enough resources left in common for others and are not wasted in use. This proviso establishes

the commons as a morally protected class of resources. The morally protected class of resources is one which is in scarcity and thus cannot be wastefully appropriated by individual laborers and excluded from others (Himma 1149). The free and open source software license centers its philosophy on this idea. The conveyance of free rights to redistribute, use, modify, and so on within the free and open source license upholds the principle that all users have a moral right to the information commons, or in this case, the source code. The free and open source software movement goes one step further in applying the non-wastefulness proviso to invalidate proprietary licenses that turn the non-rivalrous nature of code into a rivalrous one that is exclusively owned by a fewer number of authors. By requiring royalties and other forms of payments, proprietary licenses exclude users from accessing what would otherwise be available for shared use, thereby wasting the information commons. The restriction of access to the commons qualifies as a form of unethical misappropriation and wastefulness. Thus, the free and open source software principle fully embraces the Lockean proviso in making source code freely and publicly available.

Despite its embodiment of the Lockean proviso, the free and open source movement faces other objections from the labor theory. One can object, for example, that authors of intellectual works in general ought to be compensated for the value they add to the product and the commons as a whole. For instance, if a software developer, albeit voluntarily, invests her time and energy into adding value to the free and open source software commons, then she should at least have the right to establish certain boundaries around her product. Legal matters aside, we can plausibly believe that the software developer is morally entitled to define such terms, so that others cannot unfairly take advantage of her contributions and exploit her labor.

One may additionally argue that if people find her contributions truly valuable or even necessary for societal well-being and human flourishing, then they will be prepared to pay for it and compensate the author.

The objection seems promising, but the free and open source software movement can address it by acknowledging one important characteristic of intellectual property. That is, authors of intellectual property are not "solely responsible for the value brought into the world by any novel piece of content" (Himma 1154). Especially in the software industry, programmers rely on nearly universal logic and common programming techniques to develop their software products. To claim that these software programmers consequently have a moral right to exclude others from their works implies the false premise that authors do not similarly appropriate the knowledge, influence, and labor of others. Thus the value-added objection fails to undermine the moral arguments of the free and open source software movement. However, it does bring to light certain caveats of the movement that cannot successfully capture all of the labor theory worries. For example, there does not seem to be a compelling moral reason to overlook the interests of the author when the intellectual product at hand is unnecessary for human well-being. If the software contributions of the author does not significantly affect social utility, then perhaps the free and open source software philosophy is too broadly applied. I will assess the plausibility of this claim in the next section.

At least in regards to the categories of socially useful software, however, the free and open source movement reasonably questions the fairness of restricting source code access on the basis of compensating laborers. Enabling software developers to privately control, patent, or protect their software means that other developers have to unnecessarily invest even more labor

and money to ensuring that they are not breaching intellectual property rights. Nonetheless, the free and open source software philosophy does recognize the importance of acknowledging the invested labor of software contributors and programmers. Its licenses at least implicitly address the worries of the labor theory in its clause on the integrity of the author's source code. The clause declares that users and programmers should appropriately acknowledge authors of earlier versions of software. Moreover, the recognition of the original authors of earlier software versions satisfies the emotional layer of compensation under the labor theory. Software programmers who feel a personal attachment to the code they develop may find that this acknowledgement is a sufficient form of compensation for their labor. Whether or not this sort of reputational preservation and recognition is enough to compensate most software developers, however, may leave something to be desired. From a normative standpoint, the free and open source software philosophy prioritizes what rights users ought to have, and only considers developers' rights as secondary.

In the following last section, I consider some ways the insensitivity to context may lead the free and open source software movement to overextend its philosophy. I emphasize a need to remember the different categories of software as well as software's uniqueness when deciding between the different models of free, open source, and proprietary ownership.

3.3 Other Ethical Considerations Regarding Software Use

As our discussion of the labor theory pointed out, the goal of intellectual property protection for software should be not only to protect user rights to the information common, but also to stimulate software innovation without completely disregarding the labor of software

developers. The free and open source software movement capably defends its values against objections from the labor, incentive, and social well-being theories. In fact, the free and open source software license embraces the ideas of human flourishing, innovation, and the Lockean proviso to bolster its argument. This section concludes our ethical analysis of the movement by considering other applications of the free and open source philosophy, and seeing how the movement fares in light of certain caveats. I will reintroduce points about the uniqueness of software, as well as highlight the importance of distinguishing socially valuable software based on the way it is used.

The Uniqueness of Software, Revisited

The main focus of the free and open source software philosophy is to make source code publicly available. In making source code available for free use, the movement values the uniqueness of software because open and shared use of software lends itself to interoperability. Interoperability is an integral characteristic of software that allows communication between different applications and programs. The free and open source code principle promotes the potential of software's interoperable functions by making the ideas captured by the source code more widely accessible. Programmers and consumers can thereby cooperate and share their ideas through shared platforms. The opposite effect occurs for software that is exclusively owned. Proprietary software restricts access to its inherent ideas and increases the possibility of incompatibility issues between different application and system software. In the worst case scenario, this leads to software monopolization and social disutility. Studies of healthcare software, for example, show that the lack of interoperability leads to ineffective access to patient care records between hospitals and care facilities (Barry). This unfortunately results in numerous

deaths, medical error, and ineffective care, thereby worsening societal well-being (Barry). In the best case scenario, restricting software interoperability significantly reduces the efficiency of software development because developers must redesign substitutes for copyrighted programming expressions. Over time, innovation slows down as certain developers can no longer enter and compete in the software market. The ethical principles of transparency, openness, and collaboration held by the free and open source software movement thus realize the potential of software's unique capabilities by encouraging interoperable design.

The free and open source software movement's support of software's interoperable functions is also conducive to securing the basic rights conducive to well-being. For instance, because free and open source software allows access across open networks and multitudes of devices, it can facilitate learning in educational departments, and help close the knowledge gap. Free and open source software enables educators to access, use, and redistribute lessons and various course materials easily and efficiently. Furthermore, the wide range of available learning interfaces can provide educators with the freedom to experiment with different teaching applications and resources because the free and open source license does not financially or legally limit students and teachers to one set of learning material, like a proprietary license might. This ability to use different sources of without restriction leads to a higher outcome of educational success, and therefore, improves well-being.

The philosophy of the free and open source software movement aligns with the conceptual framework of copyright better than proprietary licensing does, too. Recall that the scope of copyright protection is limited to the expressions of ideas, not the ideas themselves. When free and open source licensing terms grant people the right to access software source code,

the terms defend the copyright principle that no one can justifiably preclude others from using ideas within the source code. Conversely, proprietary software prevents others from accessing the ideas within the source code because software inherently merges idea and expression in its source code form. When proprietary software only lets users access the object code, then users are excluded from accessing the underlying ideas in addition to its expression. The simplified example of software below illustrates how the incomprehensibility of object code protects the idea, which in this case, is the software's "printing" function.

Example of Software that Prints the Statement, "Hello world!"

Source Code	Object Code

#include <stdio.h>
int main() {
printf("Hello world!");
return 0;
}
	10001100
	11010101
	00101110
	01111011
	10110100

Unlike its proprietary counterpart, the free and open source software movement makes the idea of the print statement function publicly available and accessible by granting users access to the source code. It therefore avoids the problem of copyrighting ideas rather than just the expressions of those ideas. In contrast, proprietary ownership of undisclosed source code fails to fulfill the original purpose of copyright legislation because software source code behaves in both a functional way (e.g. the printing function, "Hello world!") and in an expressive way (e.g. the literal written code). By sharing the source code, the free and open source movement implicitly complements software's unique characteristic as both intrinsically functional and expressive, and thus does not confront the issues raised by our software uniqueness discussion.

The Caveat to Free and Open Source Software

Even if the free and open source approach to software ownership is more conducive to software's unique nature and innovation, critics can point out a caveat that undermines the movement's philosophy. Specifically, the free and open source movement's reliance on alternative means of financing faces the same normative worries directed towards proprietary software ownership. Given our understanding that software can be used for socially valuable purposes, the quality of software development is critical not just for innovation, but also for human well-being. The free and open source movement's central ideology is that users should be able to freely access software code. As respectable as this commitment is from a social utilitarian standpoint, the free and open source software providers may still exploit users' rights to access and prevent them from fully maximizing the potential of socially valuable software.

This exploitation of user rights can occur when free and open source code providers purposefully leave defects in the code and then monopolize its support services. Recall that free and open source followers must use alternate means of financially supporting their developers not only to incentivize continued innovation, but also to fairly compensate them, as concerned in the labor theory. The free and open source movement earns revenue through customer services, software maintenance documentations, advertisements, and other monetary avenues. If prohibiting access to software leads to social disutility and violates user rights, then according to the same normative perspective, excluding access to support services and maintenance training is equally unjustified. Yet there is nothing in the free and open source licensing terms that explicitly commits software developers to supporting software maintenance in the long-term, upgrading earlier versions of software, or providing public and freely accessible client services

and documentation (Meyer). In fact, the license does not address the ethical worry of monopolizing such alternate financing methods in general. This should be a legitimate moral concern for the free and open source movement because without pay or royalty licensing, there is little incentive (aside from those argued in Chapter 1) for a software programmer to have a vested interest in the freedoms afforded by software that are critical to human flourishing. There could thus be a tension between the need for socially useful software and the need to properly incentive its long-term development. This problem therefore urges us to contextualize the different categories of software and apply the various ownership philosophies accordingly. Free and open source licensing schemes may be an inadequate check on the moral problems identified above, which in some cases, such as socially valuable software, may require other incentives such as government funding.

CONCLUSION

Throughout our discussion of intellectual property ownership, we emphasized the moral tension between owner and user rights. While the labor and incentive theories highlighted the necessities of author compensation, financial incentives, and exclusive control over property, we found that software embodied special functions that challenged these theories. The uniqueness of software in terms of its functionality, expressivity, as well as its use as a means to complete a task (rather than to simply express a concept) undermined the labor and incentive justifications for exclusive intellectual property ownership. Moreover I have shown that the uniqueness of software warrants a different incentivizing mechanism for software innovation and labor compensation. Preferably this new mechanism does not exclude others, and especially the least well-off, from accessing the basic rights to education, public health, agriculture, and more. The ethical worries raised by the social need for software therefore established our normative framework for evaluating different software ownership ideologies.

Among them was the free and open source software movement, which captures many of the ideas in the normative and incentive theories. It attempts to prevent and correct the exploitive practices of patent monopolization by publicly sharing the source code. Moreover, the movement establishes clear norms to facilitate an original author's attribution of effort, thereby providing some form of emotional compensation. Simultaneously, the free and open source licensing terms allow for greater collaboration, interoperability, and even alternate financing methods that do not involve blocking access to source code. Because the free and open source philosophy upholds the idea that free access to knowledge is a fundamental right, it leaves the intellectual commons

open for all to access and use. Thus although it is not without its flaws, the free and open source philosophy has the most potential for maximizing software's functionality and social utility.

In conclusion, intellectual property rights were created in order to cultivate knowledge, drive market competition, and promote meaningful innovation. The rapid development of software technology, and specifically socially useful technology, has increased the stakes of harming social well-being. However, through continued discussions amongst legislators, software developers, judges, and the greater user population, there is hope that the future of software intellectual property rights steers its course towards greater social welfare and software development.

WORKS CITED

"Assignment And Cession Of Ownership Rights (Alienation of Rights): Preparation and Registration of Agreements." *Grainconsaltex.* http://grainc.ru/en/transfer-alienation-of-rights-to-the-objects-of-intellectual-property/.

Baker, Dean; Arjun Jayadev; and Joseph Stiglitz. *Innovation, Intellectual Property and Development: A Better Set of Approaches for the 21st Century.* AccessIBSA: Innovation & Access to Medicines in India, Brazil & South Africa, July 2017, pp 1-89.

Barry, Brian. "The Emerging Role of Open Source in Healthcare." *Technology Innovation Management Review*, Nov 2008. https://timreview.ca/article/203.

Bessen, James. "Patent Thickets: Strategic Patenting of Complex Technologies." *Technology & Policy Research Initiative*, Boston University School of Law, March 2003, pp. 1-29.

Boldrin, Michele and Levine, David K. "Open-Source Software: Who Needs Intellectual Property?: IP is Not a Prerequisite for Innovation or Even Profit." *Foundation for Economic Education*, 1 January 2007. https://fee.org/articles/open-source-software-who-needs-intellectual-property/.

Brachmann, Steve. "The Top 10 Patents Issued in 2015." *IPWatchdog Institute,* 2015 December 28. https://www.ipwatchdog.com/2015/12/28/top-10-patents-issued-2015/id=64025/.

Bransco, Anne Wells. "Who Owns Information?: From Privacy to Public Access." *Perseus Books Group*, 1994, pp. 1-241.

Child, James W. "The Moral Foundations of Intangible Property." *The Monist*, 1 Oct. 1990.

Chopra, Samir. "End Intellectual Property." *Aeon*, 12 Nov 2018. https://aeon.co/essays/the-idea-of-intellectual-property-is-nonsensical-and-pernicious

"Circular 61: Copyright Registration of Computer Programs." *U.S. Copyright Office*, September 2017. https://www.copyright.gov/circs/circ61.pdf.

Collier, Roger. "Drug patents: the evergreening problem." *National Center for Biotechnology Information*, 11 June 2013. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3680578/.

"Compendium: Ch 700: Literary Works." 29 September 2017, pp. 1-51. *U.S. Copyright Office.* https://www.copyright.gov/comp3/chap700/ch700-literary-works.pdf.

Contigiani, Andrea and Hsu, David H. "How Trade Secrets Hurt Innovation." *Harvard Business Review*, 29 January 2019. https://hbr.org/2019/01/how-trade-secrets-hurt-innovation.

Dam, Kenneth W. "Some Economic Considerations in the Intellectual Property Protection of Software." *The Journal of Legal Studies*, Vol. 24, No. 2, June 1995, pp. 321-377.

Douglas, David M. "The Social Disutility of Software Ownership." *Science and Engineering Ethics*, Vol. 485, Issue 3, 2011, pp. 485-502.

Durrell, Karen Lynne. "Intellectual Property Protection for Computer Software: How Much and What Form is Effective." *International Journal of Law and and Information Technology*, Oxford University Press, October 2000, Vol. 8, Issue 3, pp 231-262.

Freibrun, Eric. "Intellectual Property Rights in Software – What They Are and How To Protect Them." *Freibrun Law*, 2 February 1995. https://freibrun.com/intellectual-property-rights-software-protect/.

Goldman, Eric. "The Problems with Software Patents (Part 1 of 3)." *Forbes*, 28 November 2012. https://www.forbes.com/sites/ericgoldman/2012/11/28/the-problems-with-software-patents/#8afb3a04391e.

Gregerson, Erik. "Patent Troll." *Encyclopedia Britannica*, 6 January 2012.

https://www.britannica.com/topic/patent-troll.

Hayman, Richard. "Human Rights Software: Information Support Solutions For Social Justice." *Information for Social Change*, Vol. 29, 2009. https://pdfs.semanticscholar.org/e000/d58ee4cda90ab36352e7a41ea7854578682b.pdf.

Hettinger, Edwin C. "Justifying Intellectual Property." *Philosophy & Public Affairs*, Vol. 18, No. 1, 1989, pp. 31-52. https://www.jstor.org/stable/2265190.

Hick, Darren Hudson. *Artistic License: The Philosophical Problems of Copyright and Appropriation*. University of Chicago Press, 26 April 2017, pp. 1-231.

Himma, Kenneth E. "The Justification of Intellectual Property: Contemporary Philosophical Disputes." *Journal of the American Society for Information and Technology*, Vol. 59, Issue 7, 2008, pp. 1143-1161.

Hoffman, Hunter G. "Virtual Reality as an Adjunctive Non-pharmacologic Analgesic for Acute Burn Pain During Medical Procedures." *National Center for Biotechnology Information*, 14 June 2015. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4465767/.

"Intellectual Property: Protection and Enforcement." *World Trade Organization*. https://www.wto.org/english/thewto_e/whatis_e/TIF_e/agrm7_e.htm.

"Intellectual Property Law: Everything You Need to Know." *UpCounsel.* https://www.upcounsel.com/intellectual-property-law.

"IP and Software." *World Intellectual Property Organization Magazine*, December 2008. https://www.wipo.int/wipo_magazine/en/2008/06/article_0006.html.

"Is Software Patentable in the United States?" *Shah IP Law, PLLC*. https://www.shahiplaw.com/software-patents/.

Karjala, Dennis S. “Oracle v. Google and the Scope of a Computer Program Copyright.” *Journal of Intellectual Property Law*. University of Georgia Law, Vol. 24, Issue 1, Article 3, October 2016. https://digitalcommons.law.uga.edu/cgi/viewcontent.cgi?referer=https://www.google.com/&httpsredir=1&article=1399&context=jipl.

James, Aaron. “When International Intellectual “Piracy” is Fair.” 2008, pp. 1-15.

McCracken, Harry. “The Ones That Didn’t Make It: Windows’ Failed Rivals.” Technologizer, 22 November 2010. https://www.technologizer.com/2010/11/22/the-ones-that-didnt-make-it-windows-failed-rivals/.

McGowan, Matthew K; Stephens, Paul; and Gruber, Dexter. “An Exploration of the Ideologies of Software Intellectual Property: The Impact on Ethical Decision Making.” *Journal of Business Ethics*, Vol 73, Issue 4, 2007, pp. 409-424.

“Method of swinging on a swing.” *Free Patents Online.* http://www.freepatentsonline.com/6368227.html.

Meyer, Bertrand. “The Ethics of Free Software.” *Software Development*, March 2000. http://se.ethz.ch/~meyer/publications/softdev/ethics.pdf.

Moore, Adam D. “Intellectual Property, Innovation, and Social Progress: The Case Against Incentive Based Arguments.” *The Hamline Law Review*, Vol. 26, 2003, pp. 602-630.

Moore, Adam and Himma, Ken. “Intellectual Property.” *The Stanford Encyclopedia of Philosophy* (Winter 2018 Edition), Edward N. Zalta (ed.),

https://plato.stanford.edu/archives/win2018/entries/intellectual-property/.

Noyes, Katherine. "10 Reasons Open Source is Good for Business." *PC World*, 5 November 2010. https://www.pcworld.com/article/209891/10_reasons_open_source_is_good_for_business.html.

Oz, Effy. "Acceptable protection of software intellectual property: a survey of software developers and lawyers." *Information & Management*, Vol 34, 1998, pp. 161-173.

Phillips, John C. "SUI GENERIS INTELLECTUAL PROPERTY PROTECTION FOR COMPUTER SOFTWARE." *George Washington Law Review*, April 1992. https://cyber.harvard.edu/property/protection/resources/phillips_unedited.html.

Reichman, Jerome H. "Intellectual Property: Does IP Harm or Help Developing Countries?" *Journal of Law, Technology & Policy*, 70-74, 2007. https://scholarship.law.duke.edu/faculty_scholarship/1927/.

"Robot action based on human demonstration." *Free Patents Online*. http://www.freepatentsonline.com/9162720.html.

Rosen, Lawrence. "Freedom and Open Source." 22 June 2004, pp. 1-12. https://rosenlaw.com/wp-content/uploads/Freedom-and-Open-Source1.pdf.

Rubinstein, Daniel. "4 successful open source business models to consider." *opensource.com*, 18 December 2017. https://opensource.com/article/17/12/open-source-business-models.

Runge, Joe. "Can You Patent an Idea?" *legalzoom*. https://www.legalzoom.com/articles/can-you-patent-an-idea.

Samuelson, Pamela. "Functionality and Expression in Computer Programs: Refining the Tests for Software Copyright Infringement." *Berkeley Technology Law Journal*, Vol 31, Issue

3, Article 3, 7 December 2017, pp. 1216-1300. https://scholarship.law.berkeley.edu/cgi/viewcontent.cgi?referer=https://www.google.com/&httpsredir=1&article=2141&context=btlj.

Sjoerdstra, Bianca. "Dealing with Vendor Lock In." *University of Twente*. https://essay.utwente.nl/70153/1/Sjoerdstra_BA_BMS.pdf.

Stallman, Richard. "The advantages of free software." *The Free Software Foundation*, 12 April 2014. https://www.gnu.org/philosophy/practical.html.

Stim, Richard. "Fair Use: What is Transformative?: In determining fair use, what makes the use of a copyrighted work "transformative"?" https://www.nolo.com/legal-encyclopedia/fair-use-what-transformative.html.

"Technology Tools in Human Rights." *The Engine Room Library*. https://library.theengineroom.org/humanrights-tech/.

"The Open Source Definition (Annotated). *Open Source Initiative*. https://opensource.org/osd-annotated.

Touretzky, David. "Source vs. Object Code: A False Dichotomy." *Carnegie Mellon University Computer Science Department*, 12 July 2000. https://www.cs.cmu.edu/~dst/DeCSS/object-code.txt.

"Trademark basics." *United States Patent and Trademark Office*, 16 February 2018. https://www.uspto.gov/trademarks-getting-started/trademark-basics.

"Trade Secret: Everything You Need to Know." *UpCounsel*. https://www.upcounsel.com/trade-secret.

"Trade Secret Protection of Computer Software." *Gesmer Updegrove,* October 1988.

https://www.gesmer.com/news/trade-secret-protection-of-computer-software.

Tuckness, Alex. "Locke's Political Philosophy." *The Stanford Encyclopedia of Philosophy* (Summer 2018 Edition), Edward N. Zalta (ed.), https://plato.stanford.edu/archives/sum2018/entries/locke-political/.

"Universal Declaration of Human Rights." *United Nations*, 10 December 1948. https://www.un.org/en/universal-declaration-human-rights/index.html.

Watson, Brett. "Philosophies of Free Software and Intellectual Property." 10 February 1999. http://www.ram.org/ramblings/philosophy/fmp/free-software-philosophy.html.

"What is free software?" *Free Software Foundation, Inc.*, 20 March 2019. https://www.gnu.org/philosophy/free-sw.html.

Wiens, Jason and Jackson, Chris. "How Intellectual Property Can Help or Hinder Innovation." *Ewing Marion Kauffman Foundation*, 7 April 2015. https://www.kauffman.org/what-we-do/resources/entrepreneurship-policy-digest/how-intellectual-property-can-help-or-hinder-innovation.

Wolf, Marty J.; Miller, Keith W; Grodzinsky, Frances S. "Free, Source-Code-Available, Or Proprietary: An Ethically Charged, Context-Sensitive Choice." *SIGCAS Computers and Society*, Vol. 39, No. 1, June 2009.

Yang, James. "Broad patents spread a wide net but more likely to be invalid." *OC Patent Lawyer*, 18 Nov 2014. https://ocpatentlawyer.com/broad-patents-spread-wide-net-likely-invalid/.