Articles from Computational Culture

Text, Speech, Machine: Metaphors for Computer Code in the Law

2012-07-26 10:07:35 matthew

As computer software has become increasingly central to commerce and creativity, lawmakers have retrofitted it into preexisting legal regimes to regulate its production and distribution. Currently in the United States, software is eligible for protection under patent law, copyright law, trade secret law and the First Amendment. Legal determinations of technology such as software do much to shape its legacy, uses and scope, as recent work in the history of patent prosecution suggests. [1] But the protean nature of computer code, which comprises software, has made it particularly challenging for lawmakers to pin down: computer code is variously treated as text, speech, or machine under US law. [2] Which legal metaphor prevails in any given case is contingent upon the particular context in which code is operating —its composer, its audience, and the nature and uses of the software it comprises. The rhetorical-legal construction of these various metaphors illustrate the complexity and flexibility of code—at once expressive, functional, political, and literary—as well as the populations lawmakers imagine to be producing and consuming code. Each of these metaphors highlights a different property of code, and together they form a set of ontologies for code.

This article examines three classificatory metaphors for computer code as they are established in the US legal system. To examine these legal metaphors for code, I draw on the social history of technology, theoretical work in judicial reasoning, and legal and cultural debates about the different kinds of intellectual property protection appropriate for computer code. [3] The tensions in code's legal definition highlights its protean nature but also its shifting cast of uses, users, and producers. Yet legal definitions do not fully determine computer code: some writers of code can and do circumvent its legal strictures. In these moments, which I visit at the end of this article, we can see how people might contest legal metaphors as a new technology is being put to new uses. The uniquely complex features of computer code, coupled with its well-documented legal history, makes it an ideal case study for how new technologies are defined and shaped by the law.

Various US court decisions, legislative records, and law review articles catalogue the construction of these metaphors and provide the central sources for my analysis. Narrowing the scope to US law is practical as well as strategic: the concentration of high-technology firms in the US as well as its position in global economics and politics has meant that US law often sets the terms for international law in intellectual property and new technology. I take a rhetorical approach to these US documents, which allows me to analyze lawmakers' assumptions about the users, producers and functions of

computer code. I begin by outlining this approach and then move to my analysis of the three central metaphors that have emerged for code in law: code as text, code as speech, and code as machine. Each of these rhetorical-legal constructions of code is an ontology of code—a snapshot of what code looks like from one perspective, extended metaphorically to its closest counterpart with legal precedence.

When new technologies meet legal precedence

Lawmakers [4] rely on precedent for regulation, so they often devise metaphors to bridge new technologies to those previously established and regulated. Sometimes these metaphors help to conveying a judicial decision figuratively, sometimes they help lawmakers reason through a new and complex technology, and sometimes they set the terms by which a new technology is governed. Whatever their function in legal discourse, metaphors can illuminate the unstable identities for technologies when they are new—before their uses become well-worn grooves through culture and communication. [5] For instance, Thomas Edison's phonograph was first classified as a 'Measuring Instrument' by the United States Patent Office in 1878 because its primary function at the time was deemed closest to the documentary function of shorthand. The phonograph pushed the limits of this patent category, however, and its later reclassification under a new 'Acoustics' patent signaled its new uses as an entertainment device. [6] The phonograph's shifting classification illustrates the way that legal metaphors for new technologies can help us chart the path a new technology takes through different functions, audiences, and means of production.

Figuratively, metaphors appear in court decisions to explain or highlight certain ideas, just as they do in academic discourse. But, in the law, metaphors can also serve as actionable language. Central metaphors for technologies can set the terms of legal discourse, initiating terms or defining them, therefore shaping the ways that subsequent subjects are treated under the law. [7] For instance, Nard, Barnes and Madison insist that 'copyright implications […] depend in part on the lawyers' and judges' selection of the "right" analogy or metaphor.' [8] These legal scholars note that although copyright law has been extended to software, copyright operates with a basic paradigm of print: 'As forms and formats have expanded, the scope of copyright law has expanded as well, but the underlying policies of the law still implicitly assume that authors and publishers, and readers follow the well-understood and established patterns of activity' (ibid., 387). Referring to this legal practice of relating new situations to established patterns of activity, Chaim Perelman and Lucie Olbrechts-Tyteca use the term 'metaphorical fusion,' a practice which 'consecrate[s] the relation between' the terms under discussion. [9] In other words, legal metaphors can move beyond the figurative bridging of ideas to determine the governance of new practices—to construct ontologies of new technologies. In our case of computer code, the metaphors of text, speech and machine that lawmakers deploy can, at least under law, render code into a text, speech or machine.

As lawmakers construct metaphors to fit code or any new technology into

preexisting paradigms, they imagine legal subjects for the new precedents or statutes they craft. They must ask: whom this law will affect? Whose activities will it promote and whose activities will it proscribe? In other words, the making of laws implicates certain users and producers—in rhetorical terms, an audience—of a new technology. [10] James Grimmelmann explains how the construction of this audience of legal subjects draws on social factors: 'Every body of law has an internal logic to it, a logic drawn from and reflected in the social relationships it imagines among the people subject to it.' [11] These socially-shaped visions often rely on some paradigmatic, 'normal' group, such as the 'average member of the public' to whom Justice Stevens referred in *Sony v. Universal*, the decision allowing users of video tape recording technology (e.g., Betamax) to record broadcasted programs. Through a concept of 'normal' people, lawmakers' imaginations of legal subjects can render some uses, users, and producers of new technologies more paradigmatic than others, thereby shaping the effective uses, users, and producers of new technologies. [12]

Ideally, at least, lawmakers aspire to write decisions and policies whose precedent will extend beyond one specific case; ideally, they aim at universality and timelessness for their rhetorical constructions of audience. Of course, no audience or rhetorical construction can be objectively universal or timeless. But, as many legal theorists have argued, the subjective construction of universality is at the heart of the judicial process.[13] This universalizing of subjective norms is the framework for Perelman and Olbrechts-Tyteca's 'universal audience,' in which the 'normal (in the law) is determined by one or more reference groups.'[14] Sometimes these hypothetical reference groups are explicitly invoked in legal code and precedent. For example, the 'reasonable person' in tort law can help determine negligence and, in patent law, the 'person having ordinary skill in the art' (PHOSITA) can help determine an invention's patentability.[15] Although these hypothetical constructions of the 'reasonable person' and the PHOSITA shift across history and context, they are nonetheless codified in the law. This legal codification of 'normal' people in the 'universal audience' can serve to naturalize certain assumptions within an argument.[16]

The concept of the 'universal audience' enables me to uncover the various political, historical, and cultural processes that were present at the genesis of each of our three legal metaphors for code because, as Antonio de Velasco argues, 'the universal audience can be seen as that always potentially contested—and thus always political—site of appeal through which truths, facts, and presumptions emerge in various contexts of symbolic production.'[17] In the legal contexts examined below, the symbolic production of laws concerning computer code reveals lawmakers' visions of their 'universal audience'—the collectivity they imagined to be subject to the new precedents they set. In other words, as lawmakers determine which precedents best fit the particular version of code under debate, they envision who can or should write code. This vision interacts with actual uses of code in complicated ways—from programmers' decisions about what code to make

open source, to uses of code in cryptography, to how companies are legally permitted to interact with each others' code—and a full examination of these implications is beyond my means here. Here I focus on the tensions between lawmakers' visions of who should write code and who actually does write code, and through three metaphors employed for code—code as text, speech, or machine—I explore disparities between the populations of those who write code and those for whom the law is written.

Computer code as text

Congress amended the 1976 Copyright Act in 1980 to define computer code as a 'literary work.' In establishing this textual metaphor for code, historically the first of our three legal metaphors, Congress was responding to recommendations from The National Commission on New Technological Uses of Copyrighted Works (CONTU). Congress commissioned CONTU in 1974 to respond to concerns of the rapidly growing software industry as well as to address mounting issues in copyright law brought about by computer technology. The US Copyright Office had permitted copyright registration for code since 1964, but this practice had not been officially recognized in copyright statute and its merits had been debated.[18] Positing code as a 'form of writing,' CONTU's Final Report recommended that copyright was, indeed, a suitable form of intellectual property protection for code.[19] By 1990, legal scholars contended that copyright for code was 'proving to be an effective and uncontroversial means for protecting program code from exact duplication' and 'virtually all nations [had] recognized the textual character of program code in deciding to use copyright law to protect it.'[20] But at the time, CONTU's recommendations were contentious among members of the commission as well as legal experts.[21] Examining the debates as this metaphor was being established can tell us much about how code was operating in US society in the mid-1970s.

Controversy centered on CONTU's recommendation that machine-readable computer code[22] should be copyrightable. That is, the controversy centered on establishing the audience for computer code.[23] CONTU's recommendation to extend copyright to works intended for machine audiences invited controversy because it departed significantly from the longstanding paradigm that copyright was only available to human-readable works—a paradigm codified in the 1909 Copyright Act. Although the 1976 Copyright Act officially overturned the 'human-readable' stipulation established in 1909,[24] the copyright of machine-readable object code was such a radical break that several subsequent cases debated its legality. Not until an appeals court ruled decisively on the issue in 1983[25] was it made clear that object code was, indeed, protected under the amended Copyright Act.[26]

In a lengthy dissent included in the CONTU *Final Report*, Commission member John Hersey, a creative writer, focused on the implications of this novel theory of machines as audience. Hersey considered object code a 'device' or a 'mechanical form' rather than a text, and warned that copyright for such a device 'forcibly wrench[es]'[27] the law, threatening to 'pollute' the stock of

existing copyrightable 'writings' and perhaps even 'blur and merge human and mechanical communication' (36). He began by highlighting this dramatic divergence from established assumptions about audience for copyrighted works:

> Works of authorship have always been intended to be circulated to human beings and to be used by them—to be read, heard, or seen, for either pleasurable or practical ends. Computer programs, in their [object code] phase, are addressed to machines. (28)

Although Hersey admitted that machine code can be read by some specially-trained programmers, he claimed that 'if a skilled programmer can "read" a program in its [...], machine-readable form, it is only in the sense that a skilled home-appliance technician can "read" the equally mechanical printed circuits of a television receiver' (30). In his explanation of object code, Hersey likened object code to a device that controls a drill and argued that few would agree to copyright protection for this mechanically-instantiated drill instruction. In contrast to Hersey's analogy, the larger Commission analogized object code to writing—albeit functional writing. Thus, both Hersey and CONTU employed metaphors for their arguments about the nature of code and its audience.
In Hersey's dissent, we see a theory of what it means to 'read' or 'write' something: 'writing' is an act of communication. Without 'reading'—by his logic, the human capability of understanding at the other end—a text cannot communicate and therefore, it cannot be 'writing.' Hersey's concern about the extension of copyright to works for machine audiences suggests that he imagined a 'universal audience' for code similar to the 'universal audience' for the literary works he produced as a creative writer; that is, he imagined only human readers as an audience. In contrast, CONTU's recommendation to treat object code as a 'literary work' implies a theory of reading and writing that includes not only humans but also machines. If machines are legitimate audiences for literary works in CONTU's formulation, then aesthetic quality and successful communication need not be relevant to the determination of copyrightability. Indeed, quoting from a case that allowed for an arbitrary string of words to be copyrighted, CONTU asserted that successful communication is not a prerequisite for copyright.[28] CONTU also invoked case law to remind readers that courts are not in the position to judge the aesthetic value of writing.[29] Thus, CONTU's *Final Report* implies a wider 'universal audience' for copyrightable works than the one imagined by Hersey.[30]
In addition to their focus on the audience for code, CONTU addressed the audience for the law itself—those who might enjoy copyright protection for their computer programs as well as those whose programs the new law might constrain. Concordant with the ends of copyright law, CONTU was explicitly interested in supporting a competitive market of creative products from a wide variety of producers.[31] Software producers were increasingly differentiating themselves from hardware producers at the time; CONTU observes this and notes that because 'program writing requires very little

capital investment,' it was possible for a very diverse marketplace for software to thrive (ibid.). An increasingly diverse marketplace meant that copyright was important for computer programs, especially for smaller operations unable to afford other means of protection (35). Copyright should be afforded to large and small firms alike, CONTU asserted, and it explicitly rejected a proposal that large firms be 'disfavored' by the new law (25). Additionally, CONTU acknowledged the end users of computer programs and recommended that they should have the right to alter programs in order to meet their needs (13). In all of these assertions, the Commission demonstrated a keen attention to the audience for the law—those who would use the new law to their advantage, whose production might be encouraged or stifled, and who might use the technologies protected by the law. Commissioner Hersey's dissent, however, reveals the lacunae in CONTU's 'universal audience' for the law. First, he is concerned with the effect on independent programmers. Although CONTU considered both small and large software firms, Hersey claimed that they envisioned these firms only in corporate contexts. He wrote,

> the picture CONTU has been given, where rights in computer programs are concerned, is that the proprietor is almost invariably corporate. If there is an individual 'author,' it will be an author for hire, whose creativity is in strict harness and whose property rights are nonexistent. (35)

Hersey went on to illustrate his argument: CONTU was aggressively lobbied by major industrial corporations; little input was sought from organizations representing independent software producers; and CONTU ignored a survey of 'smallish' firms that recommended against copyright protection for software (35). Hersey warned Congress that 'copyrighting of the machine phases of programs would be likely to strengthen the position of the large firms, to reinforce the oligopoly of these dominant companies, and to inhibit competition from and among small independents' (36). His concern for independent programmers is echoed in later debates about patent protection for software, as we will see below.

CONTU imagined the legal subjects for software copyright law as limited to those who make, circulate and use software. Hersey contended, however, that the general public is also implicated in any law, perhaps because he foresaw the wider use of software among nonspecialists. He challenged the commission by arguing that normal citizens would not understand this new application of copyright: 'Ask any citizen in the street whether a printed circuit in a microprocessor in the emission control of his or her car is a copy of a literary work, and see what answer you get' (33). The consequences for this cognitive dissonance are dire, and apparently go far beyond that of software production:

> But if our government *tells* the citizens in the street that this is so and makes it law, what then happens to the citizen's sense of distinction between works that speak to the minds and senses of

> men and women and works that run machines—or, ultimately, the citizen's sense of the saving distinction between human beings themselves and machines themselves? (33)

Put another way, Hersey warned that legal subjects could be *existentially* troubled by the granting of copyright to works intended for non-human audiences. CONTU acknowledged the potentially dehumanizing danger of computers in its *Final Report*, but contended that copyright of computer code was irrelevant to the issue (26).

Hersey and the larger Commission's different 'universal audiences' for both computer code and the copyright law that might apply to code reveal some of the political tension behind CONTU's recommendation to consider computer code—especially object code—a 'literary work.' Hersey constructs a 'universal audience' for software copyright law that includes not only the software industry but also the larger public, and he then uses this construction to reason about the audience for code. In his vision, the law's subjects would not perceive circuit diagrams as 'texts,' so Hersey's audience for code excludes machines. In contrast, CONTU sees machines as legitimate audiences for the 'writing' of computer code; works aimed at machine audiences (e.g., object code) are therefore worthy of copyright. But they imagine a narrower 'universal audience' for the law than Hersey; they picture the law's subjects only as software users and producers working primarily in corporate contexts. Congress ignored Hersey's warnings and adopted the CONTU Report's recommendations, so this corporately-inflected 'universal audience' for software copyright set the legal terms for software production in 1980. Interestingly, *Apple v. Franklin*, the 1983 decision that affirmed CONTU's recommendation that machine-readable code be subject to copyright, appeared to echo CONTU's bias toward corporate firms. The court opened their decision with a comparison of the market values of Apple versus Franklin, asserting that Apple—who won the suit—was a far larger and more successful company. The establishment of computer code as a 'literary work' in copyright law therefore initiated not only an implied legal sorting of human and machinic audiences for code but also a set of legal imbalances between corporate and independent audiences for laws regarding code.

The decision to extend copyright to code was pragmatic rather than aesthetic, but it reflected a nascent approach to code as a form of creative expression. In 1974, the same year as CONTU was formed, Donald Knuth famously declared, '[p]rogramming is best regarded as the process of creating works of literature, which are meant to be read.'[32] Code poetry, which began in the 1960s and became popular in the 1990s when it was written in Perl and other flexible, high-level programming languages, blurred the line between expression and function, dramatically enacting Knuth's assertion. Recent approaches to code through Critical Code Studies treat code as a literary work.[33] And a flourishing scene of independent programmers in the open source movement exchange code online, competing for 'elegance' and leveraging code's status as a 'literary work' under copyright law to protect their free exchange. As we will see in the next section, the *de facto* legal sanctioning of code as creative expression allowed for it to be protected

under the First Amendment.

Code as Speech

The list of activities protected as 'speech' under the First Amendment includes not only the paradigmatic oral expression, but also textual writing, methods of dress, gestures, visual art, and computer code. Excluded from speech protections are certain forms of conduct, especially if they endanger others (e.g., yelling "fire" in a crowded theater). The two series of cases addressed in this section, *Bernstein v. United States* (1996-1999) and *Universal City Studios vs. Reimerdes / Corley* (2000-2001), weigh whether code is more like speech or conduct under the First Amendment.[34] Both cases construe code as a kind of speech as well as a form of creative expression, as suggested by code's status as a 'literary work' under copyright law. In Bernstein, the trial court reasoned that because code had been allowed copyright, it had expressive elements that could be protected by the First Amendment.[35] Although copyright law worked to justify code as 'protected speech' in *Bernstein*, copyright worked against code's protection as speech in *Universal*. In *Universal*, the entertainment industry's copyright and trade secret claims trumped the 'hacker' magazine 2600's free speech claims to distribute a particular algorithm. In both cases, the chosen metaphor for code was critical for determining not only what sort of legal protections code was allowed, but also how lawmakers at the time envisioned code and how it should be allowed to circulate.

The metaphors at play in *Bernstein* tested whether cryptographic source code was 'speech,' and therefore protected under the First Amendment, or whether it was a 'defense article,' and therefore subject to regulations by the US government. Along with various kinds of weapons, high-level cryptography, because it is reserved for government communication, is classified as a 'defense article' under US State Department regulations.[36] In *Bernstein*, the State Department invoked these regulations to claim that PhD student Daniel Bernstein's cryptographic code needed to be licensed by them before he could publish it. However, Bernstein insisted that 'computer code inscribed on paper, like any non-English language, is speech protected by the First Amendment.'[37] The State Department countered that the source code for Bernstein's encryption program was not speech but conduct—which is afforded fewer legal protections—because it was functional rather than communicative (ibid.).

The District Court decided that 'source code is speech protected by the First Amendment' [38]; therefore, Bernstein had the right to circulate his algorithm as freely as he might 'speak' it. Rejecting the State Department's argument that functional speech was somehow less like speech, the court argued, 'An extension of that argument assumes that once language allows one to actually do something, like play music or make lasagne, the language is no longer speech. The logic of this proposition is dubious at best. Its support in First Amendment law is nonexistent.'[39] Here we see metaphors of code's functionality to harmless functional writing: cooking and music. When the State Department persisted in arguing that Bernstein's algorithm was not 'mere

speech,' the Court chastised them for 'assuming that the functionality of speech can somehow be divorced from the speech itself,' implying Speech Act Theory concepts about speech as action.[40] The Court reminded them that '[t]his controversy is before this court precisely because there is no clear line between communication and its consequences'(ibid.).
US courts had previously established that the language in which communication happens is immaterial to whether or not it is speech, and in *Bernstein*, they insisted that code could also be considered a kind of language: 'All [languages] participate in a complex system of understood meanings within specific communities. Even object code, which directly instructs the computer, operates as a "language."'[41] Echoing the CONTU debates as well as the debates surrounding the 1909 Copyright Act, the Court indicated that machines can participate in communicative acts:

> Like source code converted to object code, [a player piano roll] 'communicates' to and directs the instrument itself, rather than the musician, to produce the music. That does not mean it is not speech. Like music and mathematical equations, computer language is just that, language, and it communicates information either to a computer or to those who can read it. (ibid., 1435)

Although the central metaphor of speech is different, the decision in *Bernstein I* relies on an argument similar to that in the CONTU *Final Report*: writing directed at a computer is a form of 'communication,' even though it is also functional. Bernstein's metaphor of code as functional speech prevailed over the State Department's metaphor of code as conduct when the Appeals Court decided that the government's 'prepublication licensing regime challenged by Bernstein applies directly to scientific expression [...therefore] it constitutes an impermissible prior restraint on speech.'[42]
The debate over code's status as 'speech' illuminates both Bernstein's and the US State Department's construction of audiences for certain forms of code, as well as the political implications of these constructions. Bernstein imagined an audience not only of his academic peers but also of private citizens who desired high-level cryptography for their personal communication. The State Department agreed that people might want high-level cryptography for their personal communication, but they envisioned those people as potential terrorists. Therefore, the *Bernstein* court's favoring of Bernstein's construction of audience over the State Department's not only protected academic speech, but it also expanded the 'universal audience' for cryptographic code to include private citizens. The case pertains to the free speech rights of the public, the Court argued, because of cryptography's growing role in ensuring privacy and anonymity in cellular and digital communications among private citizens. The Court wrote, 'Bernstein's is a suit not merely concerning a small group of scientists laboring in an esoteric field, but also touches on the public interest broadly defined.'[43] At the heart of the decision, then, was a new concept about the role of computer code—specifically, cryptographic code—in public life. By 1999, digital communication

was sufficiently important for the Court to protect its rights of privacy through protecting the general use of high-level cryptography.

Two years later, a decision in a different case dramatically decreased these protections for the circulation of code. In direct contrast to *Bernstein*'s broad vision of speech protections for code, the court in *Universal City Studios, Inc. v. Corley* concluded, 'The functionality of computer code properly affects the scope of its First Amendment protection.'[44] The Universal cases pitted 'hackers' against the movie industry: the defendants in Universal, associated with the hacker magazine *2600*, had distributed an algorithm online called 'DeCSS' that effectively 'unlocked' a form of cryptographic anti-piracy security for DVDs.[45] The Digital Millennium Copyright Act (DMCA)—a legislative response to the entertainment industry's lobby to strengthen copyright protection—was the operative law in the case and had gone into effect just prior to *Universal*.[46] Represented by Universal Studios, the movie industry contended that the DMCA's 'anti-circumvention' and 'anti-trafficking' provisions outlawed the DeCSS code and its distribution. In accordance with the DMCA, the industry threatened websites hosting the code with 'cease and desist' letters. However, some sites—such the popular hacker magazine *2600*—left the code online, as a political statement against the DMCA and in support of the free speech protections of code established in *Bernstein*. In the arguments and conclusions in the case, litigants professed conflicting visions of how entertainment should be used and who should be using it. The entertainment industry's imposition of their narrowly constructed audience onto the more general, actual audience for copyrighted works in Universal provoked vigorous protests and contentious debates about copyright and free speech online. In these 'DeCSS protests,' people distributed the DeCSS code in forms such as haikus and images to make the point that code was a form of creative expression and should be protected as such. In so doing, they registered their rejection of the court's narrow perception of audience and use for these copyrighted works.

The admixture of speech and conduct in the DeCSS code complicated the application of the DMCA in *Universal* just as it had complicated the protection of code-based cryptography in *Bernstein*. The *Bernstein* Appeals Court had rejected the government's argument that any functionality should render expression a form of conduct because such an argument would be untenable for future digital communications: 'The fact that computers will soon be able to respond to spoken commands, for example, should not confer on the government the unfettered power to impose prior restraints on speech in an effort to control its "functional" aspects.'[47] The *Universal* courts also found it difficult to separate the expressive from the functional in code and were forced to make a decision on code as an ineluctable combination of both:

> [...] computer code can instantly cause a computer to accomplish tasks and instantly render the results of those tasks available throughout the world via the Internet. These realities of what code is and what its normal functions are require a First Amendment analysis that treats code as combining nonspeech and speech

elements, i.e., functional and expressive elements.[48]

Several prominent computer science professors testified before the court in *Universal v. Reimerdes* that although the separation of speech from non-speech elements was impossible for code, code should be governed as speech, as it had been in *Bernstein*, because its unfettered circulation was essential to research.[49]
Contrary to *Bernstein*, however, the courts in *Universal* decided that the functional aspects of the code trumped its expressive aspects. The court in *Universal v. Corley* declared that 'the capacity of a decryption program like DeCSS to accomplish unauthorized – indeed, unlawful – access to materials in which the Plaintiffs have intellectual property rights must inform and limit the scope of its First Amendment protection.'[50] Again, metaphors for the code in question were key to the decision. Here, DeCSS is a like an unauthorized key:

> CSS is like a lock on a homeowner's door, a combination of a safe, or a security device attached to a store's products. DeCSS is computer code that can decrypt CSS. In its basic function, it is like a skeleton key that can open a locked door, a combination that can open a safe, or a device that can neutralize the security device attached to a store's products. (ibid.)

Extending the allusions to theft, the court went so far as to repeat the industry's description of DeCSS as 'a digital crowbar' [ibid., 453n.28].
The metaphor changes in *Universal v. Reimerdes*, where DeCSS's resemblance to 'disease' and 'assassination' reveals the danger the court felt it posed to society. The Court wrote:

> The spread of means of circumventing access to copyrighted works in digital form, however, is analogous to a propagated outbreak epidemic. Finding the original source of infection (e.g., the author of DeCSS or the first person to misuse it) accomplishes nothing, as the disease (infringement made possible by DeCSS and the resulting availability of decrypted DVDs) may continue to spread from one person who gains access to the circumvention program or decrypted DVD to another.[51]

Under Perelman and Olbrechts-Tyteca's formulation, this metaphor fuses concepts of code and disease to suggest that code can infiltrate and corrupt the 'healthy' system of DMCA copyright protection. Intertwining the figurative disease metaphor with another metaphor of danger—assassination—the *Reimerdes* court echoes the State Department's argument about code's destructive potential in *Bernstein* and provides rhetorical proof that code can 'disable systems upon which the nation depends':

> Computer code is expressive. To that extent, it is a matter of First Amendment concern. But computer code is not purely expressive any more than the assassination of a political figure is

> purely a political statement. Code causes computers to perform desired functions. Its expressive element no more immunizes its functional aspects from regulation than the expressive motives of an assassin immunize the assassin's action. [52]

As James Boyle writes in his lucid description of *Universal v. Reimerdes*, 'it is here that the defendants lose the battle of the metaphors.'[53] These figurative metaphors direct our attention and fear to code as 'conduct' and help to build the legal metaphor for code as a form of conduct rather than speech. If code is a digital crowbar, a vector of disease, or a political assassination, it follows—at least for the courts in *Universal*—that the functional qualities of code are too dangerous to protect its expressive qualities.

One aspect of the *Universal* courts' privileging of code's function over its expression proved particularly controversial: their declaration that hyperlinking to DeCSS code was a form of 'trafficking' in 'anti-circumvention devices,' and therefore, a DMCA violation. In his opinion in *Reimerdes*, Judge Kaplan admitted that this interpretation of the DMCA's anti-trafficking provision might produce an unfortunate 'chilling effect' on expression. However, he claimed that 'the potential chilling effect of DMCA liability cannot utterly immunize web site operators from all actions for disseminating circumvention technology' (note his repeated use of the disease-verb 'immunize').[54] Judge Kaplan did not specify who a 'web site operator' might be, but given the broad access to the World Wide Web at the time, this could have included anyone who used a free online blog or put up their own webpage to post links to the Web. With this broad access, it was bold to assert that even *hyperlinking* to DeCSS code was prohibited by the DMCA. Mitigating this assertion somewhat, he declared that hyperlinkers must be at least aware of their actions to be prosecuted (ibid., 341). However, the Court tacitly assumes users of code (even through simple Web-based interfaces) have both a legal and a technical savvy that was and is still rare amongst actual Web-writers.

In these cases that construe code as a form of speech, we see several competing visions of the potential users and writers of code—that is, several 'universal audiences' of code imagined by the courts and the litigants. In *Bernstein*, the courts struck down the government's prior restraint on the scientific expression of cryptographic researchers. They also acknowledged that while the scientific audience for cryptographic source code was small and specialized, the case resonated much further. The *Bernstein* Appeals Court specifically addressed the rights of US citizens to communicate privately and indicated that, in this digital age, undue government restrictions on cryptographic code infringed on those rights. In contrast, although the Reimerdes Court admits that Linux users might deploy DeCSS for non-infringing purposes, it declares Linux's distribution to be too small to balance against the many others who would use DeCSS for infringing purposes. We might say that the *Universal* courts imagine a 'universal audience' of DVD-users who rely—or *should* rely—on only mass-market software and hardware. Ironically, the code in Bernstein—an encryption algorithm that might cloak

terrorist activities—was deemed less dangerous than the code in *Universal*—a DVD anti-encryption algorithm that had become defunct by the time of the *Corley* trial.[55] In *Bernstein*, the potential harm was directed toward the US government and private citizens, whereas in *Universal*, the potential harm was directed at the movie industry. Moreover, the economic interests of the industry seemed to figure prominently in the *Universal* courts' reasoning. The passage of the DMCA in 1998 and the terrorist attacks of 9/11/2001 might explain some of the shift in the courts' perspectives between *Bernstein* and *Universal*. But comparing the two cases, it appears that the general population's use of computer code became a central threat for the entertainment industry around the turn of the 21st century. Indeed, the decision in *A&M Records, Inc. v. Napster, Inc.*, contemporary with the decision in *Universal*, supports this claim.[56]

The prevalence of speech through digital means, especially online, makes code increasingly central to speech protections in the minds of many writers. For example, a more recent debate about free speech protection of code concerns Distributed Denial of Service (DDoS) attacks, especially those defending Wikileaks and conducted by the online force 'Anonymous.' If code is 'speech,' then are these online protests akin to angry letters and demonstrations? Or, if code is 'conduct,' then are they more like riots or sit-ins? Where does the government's compelling interest in the speech and code that we write and use verge into political censorship? In the conclusion, I revisit these ways that *Bernstein* and *Universal* serve as backdrops for contemporary questions about speech and code on the Internet as a form of political and technological protest.

Code as Machine

Programming the first computers in the early 1940s involved physically configuring the machine rather than writing text. Ironically, this first manifestation of code—code as configuration of machine—was the last to be recognized in the law. The software that code comprises is metaphorically a form of machine in patent law, the intellectual property regime that most typically pertains to manufacturing contexts. Patent law's manufacturing context is one reason patent protection for software constructs corporations as the paradigmatic writers of code.[57] The ontology of code as machine not only presupposes its producers, but it also highlights its ability to do things in the world.

The patentability of software was established not by Congress or any specific deliberation on the wisdom or feasibility of doing so, but instead through a series of cases concerning algorithms beginning in the 1980s. This case trajectory has incrementally led to a more corporate vision of the 'universal audience' for computer code in patent law. To locate the origin and potential implications of this bias toward corporate composers of code, in this section I outline the trajectory of cases and metaphors that led to patent protections for computer code. Because the specific case history is complicated and a full treatment is outside of the scope of this article, I have zoomed out to analyze the series of cases through their controlling legal

reasoning rather than their specific language.

The availability of patent protection for computer code has long been debated and often been proposed, as the CONTU *Final Report* indicated. In the 1960s, President Johnson assembled an advisory group to review the US Patent and Trademark Office's policies on software patents; the group concluded in their 1966 report that software should be specifically excluded as patentable subject matter.[58] In the 1970s, several cases concerning patents on algorithms suggested that software was unpatentable because it was too close to math, which is not patentable subject matter.[59] In the 1980s and early 1990s, various patent cases about chemical processes and 'business methods' chipped away at the assumed prohibition against 'method' patents —and, by extension, software patents. These cases indicated that any physical component could make a process patentable,[60] and that the use of the word 'machine' in a patent description rendered the described object a 'physical device'[61]—a magical transformation that gestures toward Perelman and Olbrechts-Tyteca's 'metaphorical fusion.' The decision that finally confirmed the patentability of software in the US was *State Street Bank v. Signature Financial Group* (1998), which concerned a patent obtained on a 'business method' implemented in software. In their decision, the Federal Circuit Court insisted that any bar on the patentability of 'business methods' (and, by metaphorical extension, software) was a misunderstanding, and they 'take this opportunity to lay this ill-conceived exception to rest.'[62] Patents for software had been granted before the decision in *State Street*, but their official sanctioning by the court opened the floodgates on applications.[63]

Patents require extensive legal knowledge and expense to obtain, which makes them more favorable for larger corporations. Unlike copyright, which is granted to a qualifying work as soon as it is 'fixed,' a US patent must be acquired through an application to the US Patent and Trademark Office (USPTO). Patents may be obtained on 'any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof' (35 U.S.C. §101) as long as it is 'novel' (§102) and 'nonobvious' (§103). Patent applicants must disclose sufficient detail on their invention to 'enable any person skilled in the art to which it pertains [the 'PHOSITA'][64]. . .to make and use the same' (35 U.S.C. §112). The disclosure ostensibly gives information about the invention to the public, who may practice the invention once the limited monopoly expires. It is more expensive and difficult to acquire a patent than a copyright, but patents allow narrower exceptions for fair use and stronger protections against infringement. These more robust patent rights are balanced with a shorter duration than copyright: a US patent lasts for only 20 years.[65]

Because the operating metaphor for code in patent law is *machine*, software's functionality is foregrounded as its dominant characteristic. The machine metaphor for software also figured prominently in a 'Manifesto Concerning the Legal Protection of Computer Programs' published by several influential legal theorists and technologists prior to the legal sanctioning of patents for software in *State Street*. Prefiguring the speech vs. conduct debate

about code in *Bernstein*, Samuelson, et al. argue that text is an ill-fitting metaphor for software because code 'behaves'. They go on to suggest 'machine' as a better metaphor:

> Traditional literary works, such as books, do not behave. Programs, like other machines, do. Programs have a *dual* character because they are textual works created specifically to bring about some set of behaviors. We attempt to capture this intriguing dual nature of software by describing software as a machine whose medium of construction happens to be text.[66]

The authors indicate that the establishment of an apt legal metaphor for software is necessary so that its protection under the law will fit software's uses, behavior, and market value. But the tensions implied in software's dual nature as text and machine renders those both of those legal metaphors individually inadequate. Rather than retrofitting software into established legal metaphor, then, the authors propose a new (*sui generis*) form of protection specific to software. Congress has not opted to adopt their proposal, however; nor have they responded fully to the US Supreme Court's more recent suggestion to review the patent system for technologies such as software.[67] If and when they do, the highly controversial machine metaphor for software may be revised. Until then, patent applications for software rely heavily on calling software a 'machine,' thus legally, rhetorically, and somewhat magically turning their claimed software into machine.

Although the legal status of software patents status is now established in the US, it is heavily critiqued. Many have argued that software patents are on shaky legal ground due to their lack of efficacy for encouraging innovation,[68] poor quality,[69] inability to distinguish between patentable and nonpatentable algorithms,[70] and bias toward larger companies.[71] For our purposes, the most salient critique of software patents is that the code-writing population implied by patent law is skewed toward larger firms rather than individual or freelance programmers. The vast legal and financial resources required to obtain, wield, and defend against patents imply a relatively corporate, large, and well-funded 'universal audience' for computer code's protection under patent law. Because patent costs are not scaled to match applicants' access to resources, financial and logistical factors make it comparatively more expensive for smaller firms to obtain and wield patents to protect their intellectual property.[72] In accusations of infringement, smaller firms are also disadvantaged by their relative lack of legal knowledge. Because code-writing occurs in so many different contexts and venues, it is nearly impossible to know all possible patents that might bear on any particular software project.[73] Yet, as in many areas of the law, ignorance of patents is no defense.[74] Even if firms find the patent questionable or the accusation bogus, they will often license a patent rather than challenging it in court because it is much easier and cheaper to do so. Because of these dynamics of patent law, small firms, independent coders, and open source developers[75] cannot escape the

patent system, even if they do not file for patents themselves.
The bias toward corporate entities in patent law for software stands in contrast with the diversifying pool of people writing code.[76] Increasingly accessible programming languages, cheaper hardware, and the availability of tutorials and platforms for code on the Web have made code-writing more accessible to casual coders. These casual coders may not know that patent law governs their writing, much less know how to claim their software as 'novel' and 'nonobvious' in order to apply for a patent. In Perelman and Olbrechts-Tytecha's terms, then, the 'universal audience' assumed by patent law differs significantly from its 'concrete audience'—the actual pool of people subject to patent law regarding software. Several legal theorists have argued that this disparity between various firms' effective access to software patents indicates a broken patent system.[77] To underscore that point, independent developers widely reject patent protection for software, and often instead embrace copyright protection because it does not require funds, expertise, or registration to obtain. [78]
It is not clear if the courts deciding this trajectory of cases about algorithms for business, manufacturing, and scientific purposes were thinking about the contexts for software development. But regardless of whether courts and Congress intended to disfavor smalltime software developers through changes in patent protection, their actions indicate that they were primarily considering well-funded, well-organized corporations when making those changes. As Perelman and Olbrechts-Tyteca suggest, lawmakers need not know or intend their laws to have certain effects in order for those effects to occur. And the law can subtly shape technologies for particular groups of users, producers, and contexts even if it is does not *explicitly* name or favor those populations or contexts.

Conclusion: The unrepresented audience fights back
Each legal metaphor for code offers a different paradigm for where code can go, what it can do, and who is allowed to write and circulate it. By looking more closely at the metaphors for code, we can see some ways that the law constructs writers, distributors, and consumers of computer code. Since code now comprises much of our infrastructure for politics and expression, as well as the infrastructure for modern life through electricity grids, traffic signals, and microwaves, the 'universal audience' implied when the different legal metaphors were established may translate into who controls these infrastructures in the future.
Yet coders are not simply determined by lawmakers. Invested in what the law has to say about code, various coding communities have pushed back on lawmakers' seemingly corporatized 'universal audience' for code. Equipped with some knowledge about contract law, as well as the ability to write code that functions as a form of law by circumscribing some uses and promoting others, the 'concrete audience' for code has managed to carve space for themselves to write—if not in official laws, at least in their coding practices. Some legal subjects have protested what they perceive as a corporate bias in intellectual property laws currently being written, pointing to the fact that

heavy corporate lobbying influenced the passage of intellectual property laws such as the DMCA, the America Invents Act, and closed-door negotiations for ACTA (Anti-Counterfeiting Trade Agreement). Legal and digital protests against these laws have been particularly vigorous where they constrict the writing and distribution of computer code: for example, the Electronic Freedom Foundation has raised money, written briefs to courts, and conducted online publicity campaigns to try to fight this corporate bias in intellectual property laws concerning software patents and online music databases. We can also see resistance in the creative translations of DMCA-violating DeCSS code into poems and songs by people who perceived the banning of the DeCSS code as censorship. Other code-based protests against the corporate bias in laws about code include: open source software licensing such as the General Public License (GPL); the shuttering of Wikipedia on 18 January 2012 to protest the Stop Online Piracy Act (SOPA), then under consideration in Congress; and Distributed Denial of Service (DDoS) attacks orchestrated by the loosely networked group called 'Anonymous' in order to take down corporate websites.

I contend that Anonymous, the Electronic Freedom Foundation, and the open source community are highly aware of the political tensions in the rhetorical construction of 'universal audience' for laws, even if they do not call it that. Their resistance can be pegged, at least partially, to their perception that the laws regarding computer code were not written with them in mind. If we conceive of lawmakers implicitly constructing paradigmatic users and producers of technology through their laws, then we can see protestors of those laws as critiquing that construction, that 'universal audience.' Perelman and Olbrechts-Tyteca describe this inter-audience critique as 'audiences pass[ing] judgment on one another,' and argued that 'particular concrete audiences are capable of validating a concept of the universal audience which characterizes them.'[79] James Crosswhite explains how this critique from a *concrete* audience can undermine the perceived validity of a 'universal audience':

> If the particular audience completely rejects the universal audience constructed from it, it would weaken the argument that the conception of the universal audience is the right one. Thus, the particular audience has a role in validating the universal audience, in keeping it from losing its relation to the particular audience in question.[80]

Here, we may be reminded of CONTU Commissioner Hersey's argument that laws that go against common sense can undermine their own legitimacy. Perhaps the most successful and longstanding critique of the 'universal audience' for legal governance of code is the GPL, the software license that provides much of the legal structure for free and open source software (F/OSS). The GPL, variations of which are used for Mozilla Firefox, and the Android and Linux Operating Systems, forces those who use and modify licensed code to circulate their own code under the same terms.[81] The GPL's progenitor, Richard Stallman, claims it allows for software 'copyleft' because

it twists *copyright* law toward purposes not intended by lawmakers—purposes more aligned with the programming community to which he belongs.[82] When he created the GPL, then, Stallman was a 'concrete audience' member passing judgment on the 'universal audience' lawmakers imagined for copyright law.

The *Universal* case series prompted another concrete audience's critique that potentially undermined the legitimacy of lawmakers' 'universal audience.' Because creative users of computer code did not see their uses of code represented in the *Universal* court's assertion that code's functional qualities should trump its expressive qualities, they exercised their perceived rights to creatively express themselves in code through DeCSS distributions.[83] Anthropologist Gabriella Coleman traces the awakening understanding of the 'hacker' community (represented here by F/OSS developers) and their response to the law's infringement on their 'speech' in the DeCSS cases:

> Although Richard Stallman certainly grounded the politics of software in a liberal vocabulary of freedom, and Daniel Bernstein's fight introduced a far more legally sophisticated idea of the First Amendment for software, it was only with the DeCSS cases that a more prolific and specific language of free speech would come to dominate among F/OSS developers, and circulate beyond F/OSS proper. In the context of F/OSS development in conjunction with the DeCSS cases, the conception of software as speech became a cultural reality.[84]

As the metaphor of code as speech was consolidated through the DeCSS cases, non-corporate software communities were galvanized to protect the freedom of speech for code and other technological expressions.[85]

Are the GPL, the DeCSS protests, and other legal or code-based protests effectively invalidating lawmakers' 'universal audience' and influencing the users, producers and contexts for code against its legal definitions? Yes, in some ways. Independent coders have made space for their own coding practices even if, at times, they seem to go against the paradigms of code in the law. Code-focused legal protests have partially eroded the effectiveness of legal authority by defying it or 'hacking' it. Millions of lines of code circulate under the GPL, and the license's 'copyleft' stipulations have been successfully defended under copyright law.[86] Some code communities have developed alternative laws that better fit their needs. For instance, the Debian open source community generates pseudo-laws that help to draw them together and delineate their boundaries.[87] In other words, coding communities can use *their* laws to construct their own 'universal audience' that encompasses them as well as defines them as a community.[88] As Coleman argues, we might understand this process as 'jurisgenesis,' Robert Cover's theory of communities' construction of laws that diverge from statist interpretations of the law.[89] Official laws are always unevenly enforced—laws regarding code in particular because of legal difficulties with interpretation and jurisdiction. Moreover, since code can effectively function as law, creative coders can

essentially program their own visions of audience in their work. As a result, coding communities can, in practice, shape the ways that the law governs code, even if they cannot directly change the paradigms constructed by lawmakers.

As is often noted, code can be a form of law. But in the law, the metaphors deployed for code complicate that assertion. If code is text, then it is authored and read—by humans as well as machines—and it can be creative and expressive as well as functional. If code is speech, then it earns certain protections by the US government—at least until it verges closer to conduct that might endanger certain corporate or governmental interests. If code is machine, then its functionality can be cordoned off from its expression and monopolized under the terms of patent law. The fact that code is simultaneously text, speech and machine in US law provides a window into the complex manifestations, functions, users and uses of code in contemporary society.

References

Beauchamp, Christopher. "Who Invented the Telephone?: Lawyers, Patents, and the Judgments of History," *Technology and Culture* 51 (2010): 854–878.

Bosmaijan, Haig. *Metaphor and Reason in Judicial Opinions*. Carbondale, IL: Southern Illinois University Press, 1992.

Boyle, James. *The Public Domain: Enclosing the Commons of the Mind*. New Haven, CT: Yale University Press, 2008.

Breyer, Stephen. "The Uneasy Case for Copyright: A Study of Copyright in Books, Photocopies, and Computer Programs," *Harvard Law Review* 84 (1970): 281-351.

Burk, Dan. "Patenting Speech," *Texas Law Review* 79 (2000): 99-162.

Cardozo, Benjamin. *The Nature of the Juridicial Process*. New Haven, CT: Yale University Press, 1921.

Cohen, Julie E. and Mark A. Lemley. "Patent Scope and Innovation in the Software Industry," *California Law Review* 89.1 (2001): 1-57.

Coleman, E. Gabriella. "Code is Speech: Legal Tinkering, Expertise, and Protest among Free and Open Source Software Developers," *Cultural Anthropology* 24.3 (2009): 420-454.

——. "Three Ethical Moments in Debian." Center for Critical Analysis, Rutgers University, Working Paper Series. (2005): 1–77. Social Science Research Network. http://dx.doi.org/10.2139/ssrn.805287

Cover, Robert. *Narrative, Violence and the Law: The Essays of Robert Cover*, Eds. Martha Minow, Michael Ryan, and Austin Sarat. Ann Arbor, MI: University of Michigan Press, 1992.

Crosswhite, James. 'Universality in Rhetoric: Perelman's Universal Audience.' *Philosophy and Rhetoric* 22.3 (1989): 157-173.

de Velasco, Antonio. "Rethinking Perelman's Universal Audience: Political Dimensions of a Controversial Concept," *Rhetoric Society Quarterly* 35.2 (2005): 47-64.

Eschenfelder, Kristen R. and Anuj C. Desai. "Software as Protest: The Unexpected Resiliency of U.S.-Based DeCSS Posting and Linking," *The Information Society* 20 (2004): 101-116.

Gard, Ron W. and Elizabeth Townsend Gard, “The Present (User-Generated Crisis) is the Past (1909 Copyright Act): An Essay Theorizing the ‘Traditional Contours of Copyright’ Language,” *Cardozo Arts & Entertainment Law Journal* 28 (2011): 455-695.
Gitelman, *Lisa. Always Already New: Media, History, and the Data of Culture*. Cambridge, MA: MIT Press, 2006.
——. *Scripts, Grooves and Writing Machines: Representing Technology in the Edison Era*. Stanford, CA: Stanford University Press, 1999.
Grimmelmann, James. “The Ethical Visions of Copyright Law,” *Fordham Law Review* 77 (2009): 2005-2037.
Jaffe, Adam and Josh Lerner. *Innovation and its Discontents*. Princeton, NJ: Princeton University Press, 2004.
Klemens, Ben. *Math You Can’t Use*. Washington, D.C.: Brookings Institution, 2006.
Knuth, Donald. “Computer Programming as an Art,” *Literate Programming*. United States: Center for the Study of Language and Information, 1992.
Lessig, Lawrence. *Code and Other Laws of Cyberspace*. New York: Basic Books, 2000.
Long, Pamela O. “Invention, authorship, ‘intellectual property,” and the origin of patents: Notes toward a conceptual history,” *Technology and Culture* 32 (1991): 846–884.
Maastricht Economic and Social Research and Training Centre on Innovation and Technology, Free/Libre/Open Source Software: Policy Support Project. “Gender: Integrated Report of Findings,” 2006. Last accessed July 20, 2012. http://flosspols.org/.
MacCormick, Neil. *Rhetoric and the Rule of Law: A Theory of Legal Reasoning*. Oxford, UK: Oxford University Press, 2005.
Marino, Mark. “Critical Code Studies,” *Electronic Book Review* (4 Dec 2006): http://www.electronicbookreview.com/thread/electropoetics/codology.
Merges, Robert P. “Software and Patent Scope: A Report from the Middle Innings,” *Texas Law Review* 85 (2006): 1627–1676.
Nard, Craig A., David W. Barnes, and Michael J. Madison. *The Law of Intellectual Property*. New York: Aspen Publishers, 2006.
Perelman, Chaim and Lucie Olbrechts-Tyteca. *The New Rhetoric*. South Bend, IN: University of Notre Dame Press, 1969.
Samuelson, Pamela. “Benson revisited: The case against patent protection for algorithms and other computer program-related inventions,” *Emory Law Journal* 39 (1990): 1025-1401.
——. “Symposium: The Future of Software Protection: Introduction,” *University of Pittsburgh Law Review* 47.4 (1986): 905-906.
——. “The Uneasy Case for Software Copyrights Revisited,” *The George Washington Law Review* 79 (2012): 1746–1782.
——. “Why Copyright Law Excludes Systems and Processes from the Scope of Its Protection,” *Texas Law Review* 85 (2007): 1921-1977.
Samuelson, Pamela, Randall Davis, Mitchell D. Kapor, and J. H. Reichman, “A Manifesto Concerning the Legal Protection of Computer Programs,” *Columbia Law Review* 94.8 (1994): 2308-2431.
Software Freedom Law Center. “BusyBox Developers and Xterasys

Corporation Agree to Settle GPL Lawsuit" (17 Dec 2007). Last accessed July 20, 2012. http://www.softwarefreedom.org/news/2007/dec/17/busybox-xterasys-settlement/.
Stallman, Richard. "Copyleft: Pragmatic Idealism," GNU Operating System. Last modified June 10, 2012. http://www.gnu.org/philosophy/pragmatic.html .
Strasser, Matthias. "A New Paradigm in Intellectual Property Law? The Case Against Open Sources," *Stanford Technology Law Review* 4 (2001): 2-70.
Swanson, Kara. "The Emergence of the Professional Patent Practitioner," *Technology and Culture* 50 (2009): 519–548.
Tien, Lee. "Publishing Software as a Speech Act," *Berkeley Technology Law Journal* 15 (2000), Last accessed July 19, 2012. http://www.law.berkeley.edu/journals/btlj/articles/vol15/index.htm.
Touretzky, David S. "Gallery of CSS Descramblers." Last modified February 13, 2008. http://www.cs.cmu.edu/~dst/DeCSS/Gallery/index.html .
United States National Commission on New Technological Uses of Copyrighted Works (CONTU), *Final Report of the National Commission on New Technological Uses of Copyrighted Works*. 31 July 1978, Washington, D.C.: Library of Congress. Last accessed June 18, 2012. http://people.ischool.berkeley.edu/~bcarver/mediawiki/images/8/89/CONTU.pdf
Vee, Annette. "Carving up the Commons: How Software Patents are Impacting our Digital Composition Environments," *Computers and Composition* 27 (2010): 179-192.
Weber, Steven. *The Success of Open Source*. Cambridge, MA: Harvard University Press, 2004.

Acknowledgements
This work benefited from comments from Kate Vieira, Tim Laquintano, Michael Bernard-Donals, Don Bialostosky, Dan Morgan, James Jasinski, and several anonymous reviewers, and I thank them for their time and attention. Thanks also to the Computational Culture editorial collective for their support during the publication process.

1. See Christopher Beauchamp, 'Who Invented the Telephone?: Lawyers, Patent and the Judgments of History,' *Technology and Culture* 51 (2010): 854–878, and Kara Swanson, 'The Emergence of the Professional Patent Practitioner,' *Technology and Culture* 50 (2009): 519–548. (up)
2. Introducing a special issue of a law review journal that focused on the legal protection of software, Pamela Samuelson wrote, 'Because of the unique character of software—part 'writing,' part 'machine'—a host of complex problems have been cropping up concerning the extent to which existing forms of intellectual property law can be adapted to accommodate software.' Pamela Samuelson, 'Symposium: The Future of Software Protection: Introduction,' *University of Pittsburgh Law Review* 47.4 (1986): 905-906, 905. (up)

3. For leading work in the field of intellectual property and computer code, see work by legal scholars Pamela Samuelson, Dan Burk, Robert P. Merges, Mark A. Lemley, and Julie E. Cohen. Relevant law review essays include: Julie E. Cohen & Mark A. Lemley, 'Patent Scope and Innovation in the Software Industry,' *California Law Review* 89.1 (2001): 1-57 (discussing the various Federal Circuit decisions that led to the patentability of algorithms); Robert P. Merges, 'Software and Patent Scope: A Report from the Middle Innings,' *Texas Law Review* 85 (2006): 1627–1676 (reviewing the impact of crucial decisions several years earlier that made software patentable); Pamela Samuelson, 'Why Copyright Law Excludes Systems and Processes from the Scope of Its Protection,' *Texas Law Review* 85 (2007): 1921-1977 (arguing for the precedence of Baker v. Selden and subsequent cases that delineate idea from expression); Pamela Samuelson, 'The Uneasy Case for Software Copyrights Revisited,' *The George Washington Law Review* 79 (2012): 1746–1782 (revisiting and updating Stephen Breyer's 1970 Harvard Law Review article in which he debates the merits of allowing copyright for software); Dan Burk, 'Patenting Speech,' *Texas Law Review* 79 (2000): 99-162 (demonstrating that software is speech and also patentable, which may open the door for other forms of patented speech); Matthias Strasser, 'A New Paradigm in Intellectual Property Law? The Case Against Open Sources,' *Stanford Technology Law Review* 4 (2001): 2-70 (reviews patent, copyright, and trade law protections for software, especially regarding the distinctions between source code and object code). (up)
4. I use 'lawmakers' here and throughout to refer to legislators, judges, courts, congressional committee members, and other people involved in recommending, making, and shaping laws in a broad sense. In each case study below, I specify which group of lawmakers are under discussion. (up)
5. See Lisa Gitelman for a review of how 'when media are new, they offer a look into the different ways their jobs get constructed as such' (6). *Always Already New: Media, History, and the Data of Culture* (Cambridge, MA, 2006). (up)
6. Lisa Gitelman, *Scripts, Grooves and Writing Machines: Representing Technology in the Edison Era* (Stanford, CA, 1999): 97-98. Gitelman reminds us that patents process both old and the new technology together, as they describe new technology using references to 'prior art.' Patents in new areas, such as Edison's phonograph, need to be fit into old classifications, at least until new classifications are devised. (up)
7. Chaim Perelman and Lucie Olbrechts-Tyteca, *The New Rhetoric* (South Bend, IN, 1969): 397. A vast body of literature exists on the use of metaphors and narrative in legal reasoning, although Perelman and Olbrechts-Tyteca's work has been useful to rhetorician studying metaphors in the law. Drawing on Perelman and Olbrechts-Tyteca, Haig Bosmajian traces the ways metaphors can turn into doctrine as they are repeated in judicial decisions, for example in the development of the 'chilling effect' trope. Bosmajian's exploration of metaphor has been

central to my thinking here. Haig Bosmajian, Metaphor and Reason in Judicial Opinions (Carbondale, IL, 1992). (up)

8. Craig A. Nard, David W. Barnes, and Michael J. Madison, *The Law of Intellectual Property* (New York, 2006): 387-88. (up)
9. Perelman and Olbrechts-Tyteca, *The New Rhetoric*, 401. (up)
10. There are at least two audiences implied in the making of laws for new technologies: the audience for the law itself, that is, lawyers and other lawmakers; and the audience for the implications of the law—the people imagined as the users and producers of new technologies. Although lawyers and other lawmakers interpret and therefore continue to shape laws, the latter audience is the more interesting for examining the perceived functions of new technologies. Consequently, this study focuses on the audience that lawmakers imagine to be subject to the law, rather than the lawyer/lawmaker audience for the legal texts themselves. (up)
11. James Grimmelman, 'The Ethical Visions of Copyright Law,' *Fordham Law Review* 77 (2009): 2005-2037, 2005. (up)
12. Here is one relevant section in the Court's opinion, written by Justice Stevens: 'the average member of the public uses a VTR (video tape recorder) principally to record a program he cannot view as it is being televised and then to watch it once at a later time. This practice, known as 'time-shifting,' enlarges the television viewing audience. For that reason, a significant amount of television programming may be used in this manner without objection from the owners of the copyrights on the programs' (421). Sony Corp. of America v. Universal City Studios, Inc., 464 U.S. 417 (1984). (up)
13. See the influential legal theorist (and later Supreme Court Justice) Benjamin Cardozo, *The Nature of the Juridicial Process* (New Haven, 1921) for an early discussion of this theory in jurisprudence: 'the traditions of our jurisprudence commit us to the objective standard. I do not mean, of course, that this ideal of objective vision is ever perfectly attained. We cannot transcend the limitations of the ego and see anything as it really is. None the less, the ideal is one to be striven for within the limits of our capacity' (106). More recent echoes of this theory of the law can be found in Neil MacCormick, *Rhetoric and the Rule of Law: A Theory of Legal Reasoning* (Oxford, UK, 2005). MacCormick argues that 'universalization is essential to justification within practical reasoning' (78) but 'objective standards are applicable only through adjudicative subjectivity' (164). (up)
14. Perelman and Olbrechts-Tyteca, *The New Rhetoric*, 73. (up)
15. The PHOSITA is an expert in the given area where the proposed patent operates; if a PHOSITA would recognize the proposed patent as part of the normal practices of the profession, then it is not 'novel' and is therefore unpatentable, see 35 U.S.C. § 103(A). Through common law, a similar hypothetical person is implied in torts, where the expectations and knowledge of a 'reasonable person' (historically, a 'reasonable man') are integral to determining negligence. A 'reasonable person' would not, for instance, throw a heavy object out of a third floor window

overlooking a busy sidewalk; therefore, anyone who did that would be legally negligent. (up)

16. Antonio de Velasco, 'Rethinking Perelman's Universal Audience: Political Dimensions of a Controversial Concept,' *Rhetoric Society Quarterly* 35.2 (2005): 47-64, 54. (up)
17. Ibid., 51. (up)
18. The merits of copyright for software were perhaps most influentially weighed by Stephen Breyer, 'The Uneasy Case for Copyright: A Study of Copyright in Books, Photocopies, and Computer Programs,' *Harvard Law Review* 84 (1970): 281-351. (up)
19. United States National Commission on New Technological Uses of Copyrighted Works (CONTU), *Final Report of the National Commission on New Technological Uses of Copyrighted Works*, (31 July 1978, Washington, D.C.: Library of Congress). Hereafter 'CONTU Report.' (up)
20. Pamela Samuelson, Randall Davis, Mitchell D. Kapor, and J. H. Reichman, 'A Manifesto Concerning the Legal Protection of Computer Programs,' *Columbia Law Review* 94.8 (1994): 2308-2431, 2429. (up)
21. Because the commission was structured primarily to address databases and photocopying rather than intellectual property protection for computer code, Pamela Samuelson argues that it was not well-prepared to address the highly technical implications of copyright for various forms of code ('CONTU Revisited'). (up)
22. There are two primary audiences for computer code—human programmers and the computer itself; 'source code' is what programmers generally write and read, and 'object code' is what computers generally write and read. Source code includes English words like ADD and JUMP, rendering its odd poetry a plausible case for code as a 'literary work.' In contrast, object code—machine-readable code—is a series of 1's and 0's. (up)
23. Samuelson, 'CONTU Revisited'; Samuelson, 'The Uneasy Case,' 1748. (up)
24. The 1976 Copyright Act declared that 'Copyright protection subsists, in accordance with this title, in original works of authorship fixed in any tangible medium of expression, now known or later developed, from which they can be perceived, reproduced, or otherwise communicated, either directly or *with the aid of a machine or device*' (17 U.S.C. §102(a), emphasis added). For further discussion of this break with tradition established in *White-Smith v. Apollo* (209 U.S. 1. (1908), concerning player piano rolls) and the 1909 Copyright Act, see Ron W. Gard and Elizabeth Townsend Gard, 'The Present (User-Generated Crisis) is the Past (1909 Copyright Act): An Essay Theorizing the 'Traditional Contours of Copyright' Language,' *Cardozo Arts & Entertainment Law Journal* 28 (2011): 455-695. (up)
25. Apple Computer, Inc. v. Franklin Computer Corp. 714 F. 2d 1240. (3rd Cir. 1983). (up)
26. Merges, 'Software and Patent Scope.' (up)
27. CONTU Report, 31. (up)
28. CONTU Report, 14. The case they used to illustrate this point was Reiss

v. National Quotation Bureau, Inc. 276 Fed. 717, 719. (S.D.N.Y. 1921). (up)
29. CONTU Report, 25. The case they invoked was Bleistein v. Donaldson Lithographing Co. 188 U.S. 239 (1903). (up)
30. The metaphor of code as functional writing and a disregard for its communicative effects are echoed in the Appeals Court decision in *Apple v. Franklin*, the case that affirmed the copyrightability for object code. Confirming Apple's copyright in computer operating system programs, the Appeals Court decision states that 'the category of 'literary works' (…) is not confined to literature in the nature of Hemingway's *For Whom the Bell Tolls*' (Apple v. Franklin at 1249). Both CONTU's *Final Report* and the decision in *Apple v. Franklin* consciously echo *Bleistein v. Donaldson Lithographing Co.*, a 1903 case that suggested a wide potential audience for creative works: 'if they command the interest of any public, they have a commercial value (…) and the taste of any public is not to be treated with contempt' (252). CONTU and the *Apple v. Franklin* Appeals Court indicate that computers are another type of audience 'not to be treated with contempt.' (up)
31. CONTU Report, 11. (up)
32. Donald Knuth, 'Computer Programming as an Art,' *Literate Programming*, (United States: Center for the Study of Language and Information, 1992): ix. (up)
33. Mark Marino, 'Critical Code Studies,' *Electronic Book Review* (4 Dec 2006), http://www.electronicbookreview.com/thread/electropoetics/codology . (up)
34. Both cases have a complicated procedural history involving several appeals and shifting plaintiffs and defendants, but the broad perspective of this article as well as the general consistency in the trajectory of decisions leads me to refer to each case as a relatively coherent block. When I use '*Bernstein*,' or '*Universal*,' I am referring to the series of cases as a whole. (up)
35. Burk, 109; Bernstein v. United States Department of State, F. Supp. 1426 (N.D. Cal. 1996) at 1436. Hereafter '*Bernstein I*.' (up)
36. Bernstein's algorithm was first constrained under International Trafficking in Arms Regulations (ITAR); due to federal regulatory changes, later *Bernstein* cases refer to Export Administration Regulations (EAR). (up)
37. *Bernstein I*, 1434. (up)
38. Bernstein v. United States Department of State, F. Supp. 1288 (N.D. Cal. 1997) at 1303. Hereafter '*Bernstein III*.' (up)
39. *Bernstein I*, 1436. For more on the application of Speech Act Theory to software, see Lee Tien, who clarifies that we should not be asking, 'is software speech?' but instead, 'Is Bernstein speaking?' (n.p.). Tien explores the difference between the two questions through Speech Act Theory's illocutionary force. 'Publishing Software as a Speech Act,' *Berkeley Technology Law Journal* 15 (2000). (up)
40. *Bernstein III*, p 1303. (up)
41. *Bernstein I*, 1436. (up)
42. Bernstein v. United States Department of Justice, 176 F.3d 1132 (9th Cir.

1999) at 4244. Hereafter, '*Bernstein v. US DoJ*.' (up)
43. Ibid., 4242. (up)
44. Universal City Studios, Inc. v. Corley, 273 F.3rd 429 (2nd Cir. 2001) at 452. Hereafter '*Corley*.' In this section, I also focus on the appeal to that case, Universal City Studios, Inc. v. Reimerdes, 111 F. Supp. 2d 294 (S.D.N.Y. 2000). Hereafter '*Reimerdes*.' (up)
45. In 2000, Universal City Studios sued Eric Corley, editor of the magazine *2600*, for 'trafficking' in the DeCSS algorithm, which circumvented the movie industry's CSS (Content Scramble System) anti-piracy algorithm. In 1999, a teenaged Norwegian named Jon Johansen had written DeCSS ostensibly so that he could play DVDs on his Linux-based computer, although the algorithm also enabled users to pirate DVDs. He shared the code online and by the end of 1999, DeCSS was ubiquitous on the Web. Although I focus here on the *Universal v. Corley/Reimerdes* case series because it came first, was higher profile, and contains the most interesting language about code as speech in the judicial decisions, there were a number of other 'DeCSS cases.' For example, in *DVD Copy Control Association v. Bunner* the DVD CCA sued Andrew Bunner and several others posting DeCSS online for misappropriation of trade secrets and won a preliminary injunction against his posting of the code as a misappropriate of trade secrets. *Bunner* went up to the Supreme Court of California, and the decision ultimately was that Bunner had a free speech right to post the code. DVD Copy Control Assn., Inc. v. Bunner, 116 Cal. App. 4th 241, 10 Cal. Rptr. 3d 185, 69 U.S.P.Q.2d 1907 (Cal. Ct. App. 2004). (up)
46. See Lawrence Lessig's discussion in of *Eldred v. Ashcroft*, a Supreme Court case that challenged the constitutionality of the DMCA, in his book *Code*. (up)
47. *Bernstein v. US DoJ*, 4235. (up)
48. *Corley*, 451. (up)
49. Touretzky, David S. 'Gallery of CSS Descramblers,' last modified Feb 13, 2008, http://www.cs.cmu.edu/~dst/DeCSS/Gallery/index.html. (up)
50. *Corley*, 453. (up)
51. *Reimerdes*, 332. (up)
52. Ibid.,, 304. (up)
53. James Boyle, *The Public Domain: Enclosing the Commons of the Mind* (New Haven, CT, 2008): 101. (up)
54. *Reimerdes*, 340. (up)
55. Kristen R. Eschenfelder and Anuj C. Desai, 'Software as Protest: The Unexpected Resiliency of U.S.-Based DeCSS Posting and Linking,' *The Information Society* 20 (2004): 101-116. (up)
56. A&M Records, Inc. v. Napster, Inc., 239 F.3d 1004 (9th Cir. 2001), affirming, 114 F.Supp.2d 896 (N.D. Cal. 2000). See also MGM Studios, Inc. v. Grokster, Ltd. 545 U.S. 913 (2005). (up)
57. Many legal scholars have noted this assumption, including Jaffe and Lerner, *Innovation*, and Ben Klemens, *Math You Can't Use*. (Washington, D.C., Brookings Institution, 2006). (up)
58. Samuelson, 'CONTU Revisited.' (up)

59. Gottschalk v. Benson, 409 U.S. 63 (1972); Parker v. Flook, 437 U.S. 584 (1978). (up)
60. Diamond v. Diehr, 450 U.S. 175 (1981). (up)
61. In re Alappat, 33 F.3d 1526 (Fed. Cir. 1994). (up)
62. State Street Bank and Trust v. Signature Financial Group, 149 F.3d 1368 (Fed. Cir. 1998). Hereafter '*State Street*.' (up)
63. Merges, 'Software and Patent Scope.' (up)
64. The PHOSITA is a hypothetical person, constructed to match each case. See my earlier discussion about normative populations in the law. (up)
65. For an excellent social history of patent law, see Pamela O. Long, 'Invention, authorship, 'intellectual property,' and the origin of patents: Notes toward a conceptual history,' *Technology and Culture* 32 (1991): 846–884. (up)
66. Samuelson, et al., 'A Manifesto,' 2320. (up)
67. Bilski et al. vs. Kappos. Bench Opinion. 561 U.S._____ . (2010), 10. *Bilski* offered hope to critics of software patents, but the Supreme Court's narrow ruling and passing the buck to Congress dashed those hopes. (up)
68. Jaffe and Lerner, *Innovation*. (up)
69. Ibid. (up)
70. Pamela Samuelson, '*Benson* revisited: The case against patent protection for algorithms and other computer program-related inventions,' *Emory Law Journal* 39 (1990): 1025-1401. (up)
71. Lessig, *Code*. (up)
72. Ibid. (up)
73. Patent infringement can occur even when the accused infringer was not aware of the patented original. This is a critical difference from copyright, in which access to the original must be established in order to prove infringement. See Klemens, *Math You Can't Use*, on the wide variety of software-writing venues. (up)
74. The decision in Lough v. Brunswick Corp., 86 F.3d 1113 (Fed. Cir. 1996), which invalidated a patent on an outboard motor component for an individual inventor because of his failure to follow a bureaucratic patent requirement, indicated that inventors are subject to the same stipulations for patent applications regardless of their knowledge or level of resources. (up)
75. The patent application process 'tilt(s…) to harm open code developers,' according to Lessig, because of the difficulty of naming 'inventors' for patents or gathering funds amongst developers in a loosely-organized, often Internet-based project such as F/OSS development (Lessig, *Code*, 213). (up)
76. Annette Vee, 'Carving up the Commons: How Software Patents are Impacting our Digital Composition Environments,' *Computers and Composition* 27 (2010): 179-192. (up)
77. Jaffe and Lerner, *Innovation*. (up)
78. Vee, 'Software Patents'. (up)
79. *The New Rhetoric*, 35. (up)

80. James Crosswhite, 'Universality in Rhetoric: Perelman's Universal Audience,' *Philosophy and Rhetoric* 22.3 (1989): 157-173, 167-8. (up)
81. See Weber, *Success of Open Source*, for more on the history of the GPL and open source software. (up)
82. Richard Stallman, 'Copyleft: Pragmatic Idealism,' *GNU Operating System*, last modified Jun 10, 2012, http://www.gnu.org/philosophy/pragmatic.html. (up)
83. These online protests included the sale of t-shirts with the DeCSS algorithm printed on them, haikus that described the algorithm, instructions for implementing the algorithm in plain English, and even recorded musical versions of the code. In his website that catalogues these playful and problematic DeCSS distributions, David S. Touretzky, a computer scientist who testified in *Reimerdes*, points to this difficulty of separating form from function in code: 'If code that can be directly compiled and executed may be suppressed under the DMCA (…) but a textual description of the same algorithm may not be suppressed, then where exactly should the line be drawn?' The appellate court declined to answer this question by commenting on any of the examples that Touretsky had collected. (up)
84. E. Gabriella Coleman, 'Code is Speech: Legal Tinkering, Expertise, and Protest among Free and Open Source Software Developers,' *Cultural Anthropology* 24.3 (2009): 420-454, 438. (up)
85. In resistance events such as the DeCSS protests, Coleman goes so far as to argue that 'struggles over code (are) not only (about) hackers' productive freedom but also (about) the very meaning of democratic citizenship' ('Code is Speech,' 449). (up)
86. Software Freedom Law Center, 'BusyBox Developers and Xterasys Corporation Agree to Settle GPL Lawsuit,' Dec 17, 2007, last accessed July 20, 2012, http://www.softwarefreedom.org/news/2007/dec/17/busybox-xterasys-settlement/. (up)
87. E. Gabriella Coleman, 'Three Ethical Moments in Debian.' Center for Critical Analysis, Rutgers University, Working Paper Series. (2005): 1–77. (up)
88. These audiences are also, of course, exclusionary, and sometimes in very problematic ways. Critiques of the social structure of free and open source software development are often focused on issues of gender, as free and open source software developers are overwhelmingly male. For an excellent and comprehensive report on gender in free and open source software, see Maastricht Economic and Social Research and Training Centre on Innovation and Technology, Free/Libre/Open Source Software: Policy Support Project, 'Gender: Integrated Report of Findings,' (Maastrict, 2006), http://flosspols.org/. (up)
89. Robert Cover, *Narrative, Violence and the Law: The Essays of Robert Cover*, Eds. Martha Minow, Michael Ryan, and Austin Sarat (Ann Arbor, MI, 1992), 103-113. (up)

Series Navigation

<< Heterogeneous Software Engineering: Garmisch 1968, Microsoft Vista, and a Methodology for Software StudiesWhat is in PageRank? A Historical and Conceptual Investigation of a Recursive Status Index. >>