THE RISE OF THE INFORMATION PROCESSING PATENT

BEN KLEMENS*

ABSTRACT

Now is the right time to revisit an old but still contentious question: should software and business methods be patentable? On the legal front, the debate hinges over whether a claimed invention consisting of an information processing step plus a trivial physical step can be claimed as a bona fide physical innovation. The Supreme Court ruled three times that it can not, but the Court of Appeals for the Federal Circuit ruled that the combination must be considered as a whole, meaning that it is to be treated like any other physical invention. On the economic front, there is no self-contained information processing industry: every business in every field uses software and business methods. For a multitude of reasons, patents are ill-suited for such massively decentralized industries. Therefore, this paper recommends a return to the distinction that inventions consisting of information processing plus a trivial physical step be barred from patentability.

The maelstrom over software patents and business methods has raged for a few decades now. It is an example of a more general question: where should one draw the line between what is patentable and what is not? The *Labcorp v. Metabolite* case came before the Supreme Court regarding this very question. Unfortunately, certiorari was withdrawn due to technicalities, but the dissent to the withdrawal made it clear that the question of what is patentable subject matter is a pressing one. Justice Breyer (with Justices Stevens and Souter) explains:

> Patent law seeks to avoid the dangers of overprotection just as surely as it seeks to avoid the diminished incentive to invent that underprotection can threaten. One way in which patent law seeks to sail between these opposing and risky shoals is through rules that bring certain types of

* Thanks to Tahmineh Maloney, who provided many hours' worth of commentary, critique, suggestions, and notes. Thanks also to Abigail Rudman for her deft librarianship, and John Duffy for productive debate (such as his example about contributory infringement in section II.4).

> invention and discovery within the scope of patentability while excluding others.[1]

In the case in question, the dissent indicated that the current line between the patentable and the unpatentable was not set in the right place to correctly strike this balance. Justice Breyer stated that the Court of Appeals for the Federal Circuit (CAFC), applying logic much like that discussed below, was allowing a patent to stand that hinders rather than promotes progress.[2]

With this in mind, this Article looks at the many limits to patentable subject matter that have been proposed. The weakest, and effectively the status quo, is that laws of nature may not be patented, but any application thereof may be. For example, an equation is unpatentable, but a machine that evaluates that equation is. In the case of *Labcorp*, the claim was not for a correlation but for the act of correlating. The fact that such a simple rewording can bypass the nonpatentability of laws of nature indicates that this line excludes from patentability only the purest of natural concepts. This line, which made software and business methods possible, has created a number of unresolved problems in the real world, as documented in Section I. The set of businesses that might independently invent the claims of a patent on a pharmaceutical include any business with a drug synthesis factory (a few dozen); the set of businesses that might independently invent the claims of a patent on a web site design include any business with a web site (about a hundred million). With a hundred million potential independent inventors, someone is almost certain to have written software that matches the claims of any given patent—but independent invention is not a valid defense against patent infringement claims. Thus, liability risk and opportunistic lawsuits are almost certain to appear—and in fact, they have. They are not due to details of patent procedure, but the fundamental fact that every business must process information and use business methods.

Another line, effectively encapsulated by a common reading of the Freeman-Walter-Abele test, is that bona fide physical inventions should be patentable, but information processing algorithms with a trivial physical step, such as anything that could be written to a computer file or a piece of paper,

---

[1] Labcorp v. Metabolite, 126 S. Ct. 2921, 2922 (2006) (Breyer, J., dissenting).

[2] The Supreme Court had another near-miss with the questions discussed here in Microsoft v. AT&T, 127 S. Ct. 1746 (2007). The relevant question of that case was whether a software object code can be a component of a patented invention, but we will see below that this assertion has never been under debate. The real question here is whether it can be the only novel component in a patented invention. The court made no such ruling regarding this question. In the oral arguments, Justice Breyer expressed his uneasinenss with the premise of software inventions: "I take it that we are operating under the assumption that software is patentable? We have never held that in this Court." Transcript of Record at 22, Microsoft v. AT&T, 127 S.Ct. 1746 (2007)(No. 05-1056). Regardless, because the question of the case focused on international trade issues rather than the form of Patent N discussed in this paper, the ruling did not address that form.

should not be. In his 1981 ruling in *Diamond v. Diehr*, Justice Rehnquist stated as much, acknowledging that a rubber press with a computational step was a bona fide physical device, but at the same time "insignificant postsolution activity will not transform an unpatentable principle into a patentable process."[3] Section II will show that this distinction is a much better fit to the economy at large, because the problems with applying patents to pseudo-industries such as the set of computer or business method users evaporate, as do a number of political problems.

As explained below, the CAFC rejected this line, indicating that it is impossible to bifurcate a patent into an algorithm and insignificant postsolution activity—in all cases, one must take the entire patent as a whole. If some steps are physical, then the whole is physical, and if some steps are novel and useful, then the whole is. This ruling was primarily intended to allow software—abstract instructions loaded onto a stock computer—to be patentable, but the logic immediately applies to other technically physical processes such as many business methods or the act of correlating. Knight argues that storylines—instructions written down for actors—fall into the class of patentable information designs exactly as does software—instructions written down for a computer.[4] It seems obvious that patent law should not cover storylines, as evidenced by the fact that the US Patent and Trademark Office (USPTO) has never granted a storyline patent, yet the question of codifying that intuition remains: without the requirement of an innovative physical step, how would one draw a legal line that allows the patenting of information designs such as software but does not allow the patenting of information designs such as storylines? Or has the USPTO been remiss in not granting patents on storylines and novel musical compositions?

Another possible line between the patentable and unpatentable is a technological arts test. To the consternation of tax lawyers, methods of using tax loopholes have also been patented. Would a technological arts test allow patents on methods to price shares but exclude patents to exploit tax loopholes, and is such a test optimal policy?

Section I of this paper will look at the legal context of patentable subject matter, documenting how patents on information designs transitioned from being consistently rejected by the USPTO to the current regime, where approximately one in six of the USPTO's granted patents are in information designs. Section II will look at the economic studies that have measured the effects of over a decade of software patentability, and consider the reasons why no such study has found any positive benefit.

I. THE LEGAL PERSPECTIVE

U.S. patent law is founded on the U.S. Constitution, Article I, Section 8,

[3] Diamond v. Diehr, 450 U.S. 175 (1981).

[4] Andrew F. Knight, *A Potentially New IP: Storyline Patents*, 86 J. PAT. & TRADEMARK OFF. SOC'Y 859 (2004).

Clause 8: "The Congress shall have power to 'promote the progress of Science and useful Arts, by securing for limited Times to Authors and Inventors the exclusive Right to their respective Writings and Discoveries.'"[5]

The first patent law was enacted on April 10, 1790. Thomas Jefferson, Secretary of State and gadget inventor, penned the description of what types of invention may be patented. Since then, Congress has changed only one word of Jefferson's text, replacing "art" with "process."[6] His minimally modified text is now 35 U.S.C. §101:

Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.[7]

These two items, the Constitution and 35 U.S.C. §101, are the entire text of the statutes regarding patentable subject matter. They already pose two challenges to patents on software and business methods. First, does patenting software and business methods "promote the progress of Science and useful Arts?" This question will be discussed in detail in Section I. Second, are software and business methods covered in the list of patentable subject matter described in §101?

A. Exceptions to Patentability

Looking at the statute, it may seem obvious that a computing process or business method is a process, and therefore covered. However, Congress, the courts, and even Jefferson himself understood that there are many restrictions beyond the sparse text of § 101 to what should be patented. For example, the Supreme Court's ruling in *Diamond v. Diehr* stated: "This Court has undoubtedly recognized limits to §101 and every discovery is not embraced within the statutory terms. Excluded from such patent protection are laws of nature, natural phenomena, and abstract ideas." [8] Included in the natural law exception are such things as "the heat of the sun, electricity, or the qualities of metals,"[9] while the abstract idea exception stipulates that one cannot patent "a novel and useful mathematical formula."[10] Jefferson wrote on the abstract idea exception in a letter to a colleague:

It would be curious then, if an idea, the fugitive fermentation of an individual brain, could, of natural right, be claimed in exclusive and stable property. . . . Its peculiar character, too, is that no one possesses the less, because every other possesses the whole of it. He who receives an idea from me, receives instruction himself without lessening mine; as he who lights his

[5] U.S. CONST. ART. I, § 8, CL. 8.

[6] *See* Diamond v. Chakrabarty, 447 U.S. 303, 309 (1980).

[7] 35 U.S.C. § 101.

[8] Diamond v. Diehr, 450 U. S. at 185.

[9] Funk Bros. Seed Co. v. Kalo Inoculant Co., 333 U. S. 127, 130 (1948).

[10] Parker v. Flook, 437 U.S. 584, 585 (1978).

taper at mine, receives light without darkening me. That ideas should freely spread from one to another over the globe, for the moral and mutual instruction of man, and improvement of his condition, seems to have been peculiarly and benevolently designed by nature, when she made them, like fire, expansible over all space, without lessening their density in any point, and like the air in which we breathe, move, and have our physical being, incapable of confinement or exclusive appropriation. [11]

These exceptions have been repeatedly reaffirmed by the courts.[12]

Next in the list of exceptions is the mathematical-algorithm exception. From *In re Walter*: "[A] principle of nature or a scientific truth (including any mathematical algorithm which expresses such a principle or truth) is not the kind of discovery . . . which the patent laws were designed to protect."[13] This ruling indicates that mathematical algorithms express principles of nature, and thus are not man-made technologies; but whether this is so is a vehemently open question, which philosophers since Plato have been unable to resolve.

On one side of the debate are the mathematical realists.[14] Plato (taking a page from Pythagoras) saw reality as a mere reflection of pure mathematical objects, meaning that Nature is a subset of Mathematics, not the other way around.[15] Paul Erdos, one of the founders of modern network theory, was famous for stating that his results were simply copied from "The Book," a divine catalog of all mathematical results.[16]

On the other side are the formalists, who see mathematics as the manipulation of human-developed symbols that happen to sometimes reflect reality. This belief first gained a following in the mid-1700s, when Gauss and others introduced the concept of non-Euclidian geometry. In the *Elements*, Euclid asserted five axioms, and then derived other statements therefrom using logical manipulations.[17] Non-Euclidian mathematicians changed one of Euclid's axioms, and then derived a new set of results that have equal internal

[11] Letter from Thomas Jefferson to Isaac MacPherson (Aug. 13, 1813), THE WRITING OF THOMAS JEFFERSON (Albert E. Bergh & Andrew A. Lipscomb eds., 1905), *available at* http://press-pubs.uchicago.edu/founders/documents/a1_8_8s12.html.

[12] Gottschalk v. Benson, 409 U.S. 63, 71-72 (1972) (denying patentability for the discovery of a novel and useful mathematical formula); Funk Bros. Seed Co., 333 U.S. at 130 ("He who discovers a hitherto unknown phenomenon of nature has no claim to a monopoly of it which the law recognizes."); Mackay Radio & Tel. Co. v. Radio Corp. of Am., 306 U.S. 86, 94 (1939) ("[A] scientific truth, or the mathematical expression of it, is not a patentable invention . . ."); Rubber-Tip Pencil Co. v. Howard, 87 U.S. 498, 507 (1874) ("An idea of itself is not patentable . . .") ; Le Roy v. Tatham, 14 How. 55 U. S. 156, 175 (1852) ("A principle, in the abstract, is a fundamental truth; an original cause; a motive; these cannot be patented, as no one can claim in either of them an exclusive right.").

[13] *In re* Walter, 618 F.2d 758, 765 (1980).

[14] RUBEN HERSH, WHAT IS MATHEMATICS, REALLY? (1997).

[15] MORRIS KLINE, MATHEMATICS: THE LOSS OF CERTAINTY 14-18 (1980).

[16] PAUL HOFFMAN, THE MAN WHO LOVED ONLY NUMBERS 26 (1998).

[17] KLINE, *supra* note 15 at 76-88.

consistency. Mathematics suddenly went from a unified characterization of one world to being a description of many, some of which did not seem to correspond to reality.[18] Mathematics was now a game of symbol manipulation. In the words of Ludwig Kronecker: "God made the integers; all the rest is the work of man."[19] The natural philosopher Donald S. Chisum typifies the modern formalist viewpoint, asserting that "[a] mathematical or other algorithm is neither a phenomenon of nature nor an abstract concept. [A mathematical] algorithm is very much a construction of the human mind. One cannot perceive an algorithm in nature. The algorithm does not describe natural phenomena (or natural relationships)."[20] The formalist approach is a preeminently modern viewpoint. Before the early 1800s, the realist approach was the understanding of the nature of mathematics. Thus, the U.S. Constitution was written during the realist period from 500 BCE–1800 CE, so modern scholars who hope to interpret the intent of the framers of the Constitution must take care to not project onto them the neologic formalist viewpoint.

It would be futile for a brief law review article to attempt to resolve a question that has been debated for centuries, if not millennia. But when evaluating the proposed limits of patentable subject matter, the reader should note that there is such ambiguity. If a proposed limit requires for its application the resolution of an unresolvable philosophical debate, then it is probably not very practical. Nonetheless, as a matter of simple fact, mathematical algorithms were not patentable until recently. Before the rulings below, the USPTO granted only a handful of patents that could be read as mathematical algorithms, and those few could be attributed more to error or creative wording than to policy.[21] Whether mathematical algorithm patents were barred because of a natural law exception, an abstract idea exception, or a mathematical algorithm exception is immaterial. Does it make sense to have an exception for mathematical principles? It seems that many side with Plato and Erdos in contending that mathematics falls into the peculiar and

[18] The non-Euclidians were partially vindicated when Einstein showed that under some circumstances, there are non-Euclidian geometries that do a better job of describing space than Euclidian geometry does.

[19] KLINE, *supra* note 15 at 232.

[20] Donald S. Chisum, *The Future of Software Protection: The Patentability of Algorithms*, 47 U. PITT. L. REV. 959, 980-981 (1986).

[21] Stobbs cites U.S. Patent No. 3,633,176 (filed Aug. 19, 1969), a "Recursive kopy [sic] program for remote input management system," as one of the two or three earliest software patents he could find. It is a clear and obvious software patent: it is written in ALGOL, and wastes no time describing stock hardware. That is, this patent provides less grounding in hardware than any of the patents discussed below, including those struck down by the Supreme Court. This patent is so unique in the patent record, and so lacks precedent and successors, that it is hard to see its granting as anything but an error that nobody bothered to correct. Gregory Stobbs, *Information Wants To Be Free, But The Packaging Is Going To Cost You!*, 2 MICH. TELECOMM. & TECH. L. REV. 75, 76-77 (1996).

benevolent design that Jefferson discusses above. For those who see mathematics as a pure technology, I will discuss the economic motivations for the exception in Section II.

B. The Business Method Exception

There is also the question of the business method exception, typically cited as originating from *Hotel Security Checking Co. v. Lorraine Co.*[22] Examples of this exception's use seem to abound in the literature, however, as the CAFC explains in *State Street Bank & Trust Co. v. Signature Financial Group, Inc.*, a wide array of business method cases have all been rejected on either the mathematical algorithm exception or on non-novelty ground. As a result, the business method exception has never truly existed or been needed.[23] For instance, the CAFC's ruling in *In re Alappat* described two seeming business methods as unpatentable:

> *Maucorps* dealt with a business methodology [sic] for deciding how salesmen should best handle respective customers and *Meyer* involved a "system" for aiding a neurologist in diagnosing patients. Clearly, neither of the alleged "inventions" in those cases falls within any §101 category.[24]

But in *State Street*, the court clarified that "closer scrutiny of these cases reveals that the claimed inventions in both *Maucorps* and *Meyer* were rejected as abstract ideas under the mathematical algorithm exception, not the business method exception."[25]

The evidence thus seems to indicate that many judges understood there to be a business method exception, but may have erred in so doing. The *State Street* ruling clarified that most business methods, such as methods for calculating a share price, could easily be invalidated via the mathematical algorithm exception without recourse to an additional business method exception.[26] I will not attempt to resolve the question of whether such an exception has existed in the past, and will follow this ruling's lead by speaking only of the abstract idea and mathematical algorithm exceptions in the sequel.

C. Congressional Intent

The last major patent reform effort was in 1952, before software was a

[22] Hotel Security Checking Co. v. Lorraine Co., 160 F. 467 (2d Cir. 1908).

[23] State Street Bank & Trust Co. v. Signature Financial Group, Inc., 149 F.3d 1368, 1375 (Mass. 1998).

[24] *In re* Alappat, 33 F.3d 1526, 1541 (Fed. Cir. 1994); *In re* Maucorps, 609 F.2d 481, 484 (CCPA 1979); *In re* Meyer, 688 F.2d 789 (CCPA 1982).

[25] State Street Bank & Trust Co., 149 F.3d at 1376.

[26] *Id.* at 1373.

significant consideration. Accordingly, there is scant evidence in the congressional record as to whether software should be patentable. However, there is also scant evidence that Congress intended anything imaginable to be patented. For example, one Senate report regarding the 1952 reforms stated that: "[a] person may have 'invented' a machine or a manufacture, which may include anything under the sun that is made by man, but is not necessarily patentable under section 101 unless the conditions of the title are fulfilled."[27] The quotes around 'invented' and the statement that such an alleged invention is not necessarily patentable suggest that the report finds limits to what creative acts may receive patent protection.

Oddly enough, the ruling in *Diamond v. Chakrabarty* used a carefully-cropped version of this statement which gives the opposite impression: "[t]he Committee Reports accompanying the 1952 Act inform us that Congress intended statutory subject matter to 'include anything under the sun that is made by man.'"[28] But even with this reading of the act, *Chakrabarty* fails to dispute many of the above exceptions to patentability. It should be noted that as of the 1952 reform process, there was a clear understanding in the courts that there were exceptions to patentability beyond Jefferson's terse text in 35 U.S.C. §101. All of the above varieties of exception originate in pre-1952 rulings. However, the Congress failed to elucidate that the exceptions were invalid.

D. The Church-Turing Thesis

Much of the debate regarding software hinges around the definition of a mathematical algorithm, because the definition will advise how one applies the above exceptions. The general understanding is that an algorithm is a precisely-defined procedure, but that does us no good here, because it does not accommodate our intuition that the procedure for evaluating the quadratic equation we learned in high school (1. Write down the coefficients a, b, and c. 2. Find b^2. 3. Subtract 4ac. . .) differs from the algorithm for producing soap (1. Obtain sodium hydroxide, purified water, and any of a variety of fats. 2. Calculate the correct proportion of sodium hydroxide to fats, using either a saponification chart or direct calculation. 3. Gradually add the sodium hydroxide to the water, taking care to ensure that lumps do not form.). By this definition, it is no surprise that analysts such as Chisum found that "[t]he Court erred in implying that algorithms relate only to mathematical problems,"[29] because he and others were using a basically vacuous definition of the term. What patent doesn't use precisely-defined procedures?

A better definition of a mathematical algorithm would be that it is the evaluation of a purely mathematical expression. By this definition, the first procedure above is clearly a mathematical algorithm, because it enumerates the

[27] S. REP. NO. 82-2, at 2 (1952).

[28] Diamond v. Chakrabarty, 447 U.S. 303, 309 (1980).

[29] Chisum, *supra* note 20, at 976.

steps required to evaluate

$$\frac{-b\pm\sqrt{b^2-4ac}}{2a}.$$

Conversely, it would require a great deal of effort and sophistry to express the combination of materials of various types in precise configurations as the evaluation of a mathematical expression. One could write down the chemical expression for step three above,

$$H_2O+NaOH®Na2O+H_2O+H-,$$

but the real-world process of making lye water is at best an approximation of the idea presented by this expression—one must take care to ensure that lumps of lye do not form. Simply put, the quadratic equation evaluation algorithm processes information itself, while the soap making algorithm processes physical ingredients.

An important feature of mathematical algorithms not shared by physical algorithms is that any algorithm can be expressed in a literally infinite number of ways, many of which look very different to the naked eye. However, the process of running software is indeed the process of evaluating a complex expression. Donald Knuth, considered by many to be one of the founders of modern computer science, explains:

> [It is not] possible to distinguish between "numerical" and "nonnumerical" algorithms, as if numbers were somehow different from other kinds of precise information. All data are numbers, and all numbers are data. Mathematicians work much more with symbolic entities than with numbers.[30]

Different programming languages make the mathematical nature of software more or less clear. Some languages, like the STELLA modeling language, use pictures and flowcharts to describe the process, while other languages, such as APL, use expressions in notation recognizable to any mathematician. Further, a basic result of computer science known as the Church-Turing thesis indicates that an algorithm written in any language in a very large class of languages can be translated to any other—or to a pure mathematical expression.[31] That is,

[30] Letter from Donald Knuth, Professor Emeritus, to Commissioner of Patents and Trademarks, Patent and Trademark Office (Sept. 2003) (on file with author), *available at* http://lpf.ai.mit.edu/Patents/knuth-to-pto.txt.

[31] Alan M. Turing, *On Computable Numbers, with an Application to the Entscheidungsproblem*, 2 PROC. OF THE LONDON MATHEMATICAL SOC'Y 230-265 (1936); Alonzo Church, *A Note on the Entscheidungsproblem,* 1 J. SYMBOLIC LOGIC 40, 40-41 (1936); Alonzo Church, *An Unsolvable Problem of Elementary Number Theory*, 58 AM. J.

there is a mechanical means of translating any mathematical expression into a computable program, and a means of translating any computable program into a mathematical expression. From the perspective of a mathematician, then, all mathematical algorithms, regardless of whether they are expressed in English, mathematical symbols, or FORTRAN,[32] are equivalent to the mathematical expression and should therefore not be patentable under the mathematical algorithm exception. It is a direct result of the Church-Turing thesis that the patenting of pure software directly contradicts the mathematical algorithm exception.

Some non-computer scientists eye the Church-Turing thesis claim, that all software is mathematics, with suspicion. Their intuition is that there is something different between a word processor and the baroque equation that it is evaluating. For such individuals, I offer a weaker form of the Church-Turing thesis: it is impossible to write a section of the Manual of Patent Examination Procedure (MPEP) that allows the patenting of software but excludes from patentability the evaluation of purely mathematical algorithms. The proof of this is in the formal Church-Turing thesis (that software and mathematical algorithms are in the same equivalence class) and Knuth's comment that all information is data; demonstrations of this weaker Church-Turing thesis will appear repeatedly below. In short, once one type of information processing is patentable, all types are patentable. Because there are various types of information processing that many think should not be patentable, the patentability of any one type of pure information processing creates myriad problems.

E. The Supreme Court's Trilogy

But what if software is used on a physical device to produce real-world outcomes? Is a computer, upon which is programmed a mathematical algorithm, a patentable device? This is the form of the question that has been debated in the courts for decades.

Another way to cast the question is in regards to a patent with two parts:

> Patent N
> Claim 1: a useful computing machine, comprising
> (a) a mathematical algorithm, which may be creatively and painstakingly derived, but which is clearly unpatentable by the mathematical algorithm exception, and
> (b) an obvious physical step such as loading the algorithm onto a stock computer, which meets the requirements for patentable subject matter but is unpatentable because it is not novel.

MATHEMATICS 345, 345-63 (1936).

[32] FORTRAN is generally recognized as the first general-purpose programming language. Its name stands for Formula Translation, providing further indication that the engineers who wrote the language recognize the equivalence between computer code and mathematical formulæ.

Can Patent N combine these two steps to form a single device that is both novel and patentable subject matter?[33]

The Supreme Court handed down three rulings that present a rather clear perspective that an invention that includes a mathematical algorithm is not automatically excluded, but it must do something innovative beyond the algorithm to merit patentability. Simply translating the algorithm into a programming language and loading it onto a stock computer is not sufficient.

In *Gottschalk v. Benson*, the respondent sought to patent a general-purpose computer on which is loaded a method for converting from one type of binary code to another. The Supreme Court held that the presented algorithm fell soundly within the abstract idea exception discussed above.[34] The *Benson* ruling went on to quote (in concurrence) the 1966 President's Commission on the Patent System:

> Uncertainty now exists as to whether the statute permits a valid patent to be granted on programs. Direct attempts to patent programs have been rejected on the ground of nonstatutory subject matter. Indirect attempts to obtain patents and avoid the rejection, by drafting claims as a process, or a machine or components thereof programmed in a given manner, rather than as a program itself, have confused the issue further and should not be permitted.[35]

This ruling is therefore clear on reading Patent N as a mathematical algorithm with a trivial extension that only "confuse[s] the issue further," and therefore Patent N is unpatentable.[36]

The position was extended in *Parker v. Flook*, another case about a mathematical algorithm loaded onto a stock computer; this time about a computer that took a few measurements, did calculations based on the measurements, and then rang an alarm if a certain variable exceeded a certain limit.[37] There was no novelty in the invention except in the mathematical algorithm: "The only difference between the conventional methods of changing alarm limits and that described in respondent's application rests in the second step—the mathematical algorithm or formula."[38] The question in the case was thus "whether the identification of a limited category of useful, though conventional, post-solution applications of such a formula makes

---

[33] The reader will note that this formulation of the question, although difficult to resolve, is preferable to the formulations above regarding the relationship between mathematics and nature, because it does not hinge on unanswerable philosophical questions.

[34] Gottschalk v. Benson, 409 U.S. 64 (1972).

[35] *Id.* at 72.

[36] *Id.*

[37] Parker v. Flook, 437 U.S. 584, 585 (1978).

[38] *Id.* at 585-586.

respondent's method eligible for patent protection."[39] The court ruled that it does not:

> The notion that post-solution activity, no matter how conventional or obvious in itself, can transform an unpatentable principle into a patentable process exalts form over substance. A competent draftsman could attach some form of post-solution activity to almost any mathematical formula; the Pythagorean Theorem would not have been patentable, or partially patentable, because a patent application contained a final step indicating that the formula, when solved, could be usefully applied to existing surveying techniques.[40]

Again, the Court would come strongly oppose granting Patent N.

The Supreme Court repeated its position a third time in *Diamond v. Diehr.*[41] In this case, the claimed machine was not a general-purpose computer but rather a rubber-curing device that was found to be inventive itself. The steps of that patent "include installing rubber in a press, closing the mold, constantly determining the temperature of the mold, constantly recalculating the appropriate cure time through the use of the formula and a digital computer, and automatically opening the press at the proper time."[42] Because the patent covered more than just an algorithm plus a trivial physical step, it did not take the form of Patent N and the Court allowed the patent to stand. However, the ruling reiterated the above warnings regarding insignificant postsolution activity:

> [I]nsignificant postsolution activity will not transform an unpatentable principle into a patentable process. To hold otherwise would allow a competent draftsman to evade the recognized limitations on the type of subject matter eligible for patent protection. . . . Because we do not view respondents' claims as an attempt to patent a mathematical formula, but rather to be drawn to an industrial process for the molding of rubber products, we affirm the judgment of the Court of Customs and Patent Appeals.[43]

All three of these rulings take pains to make clear that "insignificant", "conventional or obvious" postsolution activity is not sufficient to transform a formula into a patentable device.[44] There are two types of insignificant

[39] *Id.* at 585.

[40] *Id.* at 590.

[41] Diamond v. Diehr, 450 U.S. 175, 175 (1981).

[42] *Id.* at 187.

[43] *Id.* at 191-193.

[44] Many, including the CAFC rulings below, seem to take *Diehr* as a reversal of the prior

postsolution activity enumerated in the discussion above. The first, in *Flook*, is the simple application of a formula to a real-world subject. The question of the case was clearly answered in the negative: a mathematical expression can not receive a patent for a limited application.[45] The second, in *Benson* and *Diehr*, is that the shrouding of a mathematical algorithm in the language of a mechanical invention will not transform it into a patentable process.[46] A consistent reading of the Supreme Court trilogy thus arrives at a moderate stance, that still excludes from patentability both pure software and a stock computer on which is loaded pure software. However, an invention that is novel and statutory subject matter but happens to rely heavily on a computer or mathematical processing is not automatically excluded from patentability.

F. The Freeman-Walter-Abele Test

Those in favor of the patentability of a stock computer on which is loaded a novel algorithm also found succor in *Diehr*, because it stated:

> In determining the eligibility of respondents' claimed process for patent protection under 101, their claims must be considered as a whole. It is inappropriate to dissect the claims into old and new elements and then to ignore the presence of the old elements in the analysis.[47]

Without the offsetting discussion above, one could interpret this to mean

two rulings. The case explicitly addresses both rulings, so if it were a true reversal, one would expect the ruling to state as much. Instead, it takes pains to explain why *Diehr* does not contradict the past rulings. The discussions of both *Benson* and *Flook* were similar; regarding *Flook*, the ruling stated: "The claims were drawn to a method for computing an 'alarm limit.' An 'alarm limit' is simply a number and the Court concluded that the application sought to protect a formula for computing this number. . . . In contrast, the respondents here do not seek to patent a mathematical formula." Diamond v. Diehr, 450 U.S. 175, 186-187.

One can interpret this two ways. The first is to assert that Justice Rehnquist was ignorant of the fact that the *Benson* and *Flook* patents were of the form of Patent N, thinking that there were no physical claims of any sort. The second is to presume that Justice Rehnquist was aware that the *Flook* invention included physical steps, but they were so insignificant that they did not merit discussion, and the claims could indeed be read as attempts to patent a formula dressed up in the language of physical registers. Those who see *Diehr* as a reversal seem to read the discussion of *Benson* and *Flook* via the first interpretation, but the second interpretation is more consistent with the distinction between significant and insignificant postsolution activity in the rest of the ruling, does not imply an unannounced reversal, and does not presume that Justice Rehnquist committed a major factual error in reading two key cases.

[45] Parker v. Flook, 437 U.S. 584 (1978).

[46] Diamond v. Diehr, 450 U.S. 175 (1981); Gottschalk v. Benson, 409 U.S. 64, 72 (1972).

[47] Diamond v. Diehr, 450 U.S. at 188.

that the dissection of Patent N into (a) an unpatentable algorithm and (b) an unpatentable stock computer is irrelevant, so long as the whole is both novel and somehow physical. But taken with the caveats and prior rulings, this passage seeks a middle ground: in some cases the 'load algorithm onto a computer' step should be detached as insignificant post-solution activity, and in others it must be considered as an integral part of the whole.

The Freeman-Walter-Abele test was the norm for deciding when an invention is of the form of Patent N and when it is an integral unit, but was eventually superseded by *In re Alappat*.[48] The thinking in *In re Freeman*,[49] *In re Walter*,[50] and *In re Abele*[51] are typified by this passage from *In re Walter*, a CCPA ruling just a year before *Diehr*:

> If the mathematical algorithm is presented and solved by the claimed invention, as was in the case in *Benson* and *Flook*, and is not applied in any manner to physical elements or process steps, no amount of post-solution activity will render the claim statutory; nor is it saved by a preamble merely reciting the field of use of the mathematical algorithm.[52]

The rule is consistent with all three Supreme Court rulings, because the application to physical elements or process steps in *Benson* and *Flook* was basically trivial, while in *Diehr*, the additional design of the rubber-curing equipment was extensive. The rule attempts to address both types of insignificant postsolution activity above: the process must be non-trivially physical, and requires more than "a preamble merely reciting the field of use."[53]

G. CAFC Rulings

The Freeman-Walter-Abele test, bolstered by its consistency with the Supreme Court rulings, was the norm in patent law for the remainder of the 1980s, but many complained that it was sometimes difficult to apply.[54] After all, any line between what is patentable and what is not patentable will have grey areas. The CAFC responded to the existence of grey areas in the Freeman-Walter-Abele test via a series of additional rulings, key among them

[48] *In re* Alappat, 33 F.3d 1526, 1543 (Fed. Cir. 1994).

[49] *In re* Freeman, 573 F.2d 1237, 1245 (C.C.P.A. 1978).

[50] *In re* Walter, 618 F.2d 758, 767 (C.C.P.A. 1980).

[51] *In re* Abele, 684 F.2d 902 (C.C.P.A. 1982).

[52] *In re* Walter, 618 F.2d at 767.

[53] *Id.*

[54] Chisum, *supra* note 20, at 992 (characterizing lower court and Patent Office decisions post-Benson as "follow[ing] a path of confusion and arbitrary distinctions," providing various examples of how different parties drew different lines between the patentable and unpatentable).

Arrhythmia Research Technology v. Corazonix Corp.,[55] *In re Alappat*,[56] *In re Lowry*,[57] *State Street Bank*,[58] and *AT & T Corp. v. Excel Communications Inc*.[59] Simply put, this series of rulings purged the Freeman-Walter-Abele test of the caveats regarding insignificant postsolution activity.

The first type of insignificant postsolution activity is prefacing an algorithm description with the phrase "a general-purpose computer on which is loaded an algorithm to." In *In re Alappat*, the CAFC ruled that

> We have held that such programming creates a new machine, because a general purpose computer in effect becomes a special purpose computer once it is programmed to perform particular functions pursuant to instructions from program software.[60]

Because programming a stock computer creates a "new machine," Patent N must not be dissected into an algorithm plus potentially insignificant postsolution activity; it is therefore fully patentable subject matter.[61] This seems to be a reversal of the Supreme Court's rulings and the Freeman-Walter-Abele test. Applying this logic to *Benson* and *Flook* would certainly allow those novel algorithms on stock hardware to stand as new machines.

The Supreme Court took the generalist approach to the question of exceptions to patentable subject matter, focusing on finding rules for patentability that, in their view, best promote the progress of Science and useful Arts throughout the economy. Conversely, the CAFC is the archetype of the specialist court, and the ruling reads as such. Only a few judges on the CAFC bench hear patent cases,[62] and as is natural, most of those are former prominent patent attorneys.[63] The fact that the CAFC's patent judges are

[55] Arrhythmia Research Technology v. Corazonix Corp., 958 F.2d 1053 (Fed. Cir. 1992).

[56] *In re* Alappat , 33 F.3d 1526 (Fed. Cir. 1994).

[57] *In re* Lowry , 32 F.3d 1579 (C.C.P.A. 1994).

[58] State Street Bank & Trust Co. v. Signature Financial Group, Inc., 149 F.3d 1368, 1373-74 (Fed. Cir. 1998).

[59] AT & T Corp. v. Excel Communications Inc., 172 F.3d 1352 (Fed. Cir. 1999).

[60] *In re* Alappat, 33 F.3d at 1545.

[61] This ruling also means that there is no such thing as a software patent, only what the European Patent Office calls a Computer Implemented Invention. This is a fiction which blurs the fact that a vast number of CIIs can be expressed in the form of Patent N, while a vast number can not. Also, it is rather linguistically difficult to maintain, so I will continue to use the term software patent to refer to patents of the form of Patent N, involving a general purpose computer on which is loaded an algorithm. Ironically, we will see below that this fiction is blocking attempts by policymakers to retain but reform software patents.

[62] John R. Allison & Mark A. Lemley, *How Federal Circuit Judges Vote in Patent Validity Cases*, 27 FLA. ST. U. L. REV. 745, 745-766 (2000).

[63] United States Court of Appeals for the Federal Circuit – Judicial Biographies, http://www.cafc.uscourts.gov/judgbios.html (The CAFC bench boasts a former member of

primarily former patent attorneys does not necessarily imply a pro-patent or anti-patent bias in specific cases, since every attorney is sometimes on the prosecution side of a patent case and sometimes on the defense. But the CAFC rulings do demonstrate a patent-oriented worldview.[64] They read like a patent attorney conversing with a client, asking whether the item in question shows creativity and effort that competitors could free-ride upon. It is notable that none of the key CAFC rulings listed above (from *Arrhythmia* to *AT&T*) use the words progress or innovation. Since every patent attorney has seen patents where a creative individual combined two common and obvious steps to form an innovative process, it is logical that the CAFC's members are reluctant to view Patent N as two separate and unpatentable steps. With regard to the above unresolvable question of natural philosophy—are mathematical results a discovery about nature or an invention of man? —one who chose a profession regarding the study of inventions is likely to lean toward seeing mathematical results as man-made; the *Alappat* ruling clearly reflects this.[65]

The second type of insignificant postsolution activity is prefacing a pure mathematical expression with a real-world subject matter. Above, *Flook* took pains to indicate that this is not sufficient to make an algorithm patentable.[66] But in *State Street*, the court ruled that an algorithm to calculate a share price, a numeric algorithm that takes in one set of numbers and outputs another via simple matrix algebra, is patentable subject matter. The ruling stated:

> The transformation of data, representing discrete dollar amounts, by a machine through a series of mathematical calculations into a final share price, constitutes a practical application of a mathematical algorithm, formula, or calculation, because it produces "a useful, concrete and tangible result"–a final share price momentarily fixed for recording and reporting purposes and even accepted and relied upon by regulatory authorities and in subsequent trades.[67]

---

the Board of directors of the American Patent Law Association, a former Chairman of the Patent Committee of the Law Section of the Pharmaceutical Manufacturers Association, a multitude of other such patent association titles, and over a hundred years of collective experience in private patent practice).

[64] ADAM B. JAFFE & JOSH LERNER, INNOVATION AND ITS DISCONTENTS: HOW OUR BROKEN PATENT SYSTEM IS ENDANGERING INNOVATION AND PROGRESS AND WHAT TO DO ABOUT IT 101 (Princeton Univ. Press 2004).

[65] *In re* Alappat, 33 F.3d at 1545 (interpreting the above Congressional statement regarding how "[a] person may have 'invented' a machine or a manufacture, which may include anything under the sun that is made by man, but it is not necessarily patentable under section 101" to indicate that Congress intended that there be no limits on patentable subject matter), *see also* Diamond v. Chakrabarty, 447 U.S. 303, 309 (1980).

[66] Parker v. Flook, 437 U.S. 584, 590 (1978).

[67] State Street Bank & Trust Co. v. Signature Financial Group, Inc., 149 F.3d 1368, 1373 (Fed. Cir. 1998).

It is very difficult to reconcile this ruling with the passage from *Flook* above that stated "the Pythagorean Theorem would not have been patentable, or partially patentable, because a patent application contained a final step indicating that the formula, when solved, could be usefully applied to existing surveying techniques."[68] It seems this is exactly the sort of thing the patent in *State Street* is doing, indicating that a series of matrix multiplications, when solved, could be usefully applied to pricing a set of mutual funds. Unfortunately, the *State Street* ruling only mentions *Flook* once in passing, so we do not know how the author of the ruling (Judge Giles Rich) would reconcile the two cases.

Again, from the narrow perspective that a business method requires creativity and effort to design, and others could free-ride on that effort, the ruling makes sense. A client walking in to an attorney's office with the *State Street* algorithm in hand would have blueprints and a story much like those of an electrical engineer client. A patent attorney hearing both stories would disdain rules dictating that one may receive a patent and the other may not. That is the specialist view; but as discussed below, a business method patent and a new circuit design are likely to have very different effects in the general economy.

The rulings make no effort to address the problem of distinguishing insignificant postsolution activity. *State Street* simply rejected the Freeman-Walter-Abele test as obsolete: "[a]fter *Diehr* and *Chakrabarty*, the Freeman-Walter-Abele test has little, if any, applicability to determining the presence of statutory subject matter."[69] It is difficult to discern whether the CAFC felt that the Freeman-Walter-Abele test was based on erroneous law or was simply out of fashion. Because the Supreme Court's caveats regarding insignificant postsolution activity have been basically ignored, patents on pure mathematical algorithms, prefaced by mention of application and loaded onto a stock computer, are routinely patented. For example:

- Patent 5,886,908: Method of efficient gradient computation
- Patent 6,434,582: Cosine algorithm for relatively small angles
- Patent 6,664,975: Cheap well-behaved affine transformation of bounding spheres
- Patent 6,721,771: Method for efficient modular polynomial division in finite fields f(2m)
- Patent 6,904,421: Methods for solving the traveling salesman

[68] Flook, 437 U.S. at 590.

[69] The word after is not to be taken as chronological, since the ordering of the rulings is *Freeman*, *Walter*, *Chakrabarty*, *Diehr*, and finally *Abele*. Thus, the Freeman-Walter-Abele test both precedes and follows *Diehr* and *Chakrabarty*. *See* State Street,149 F.3d at 1374.

problem

Inspection of all of these patents show that they are of the form of Patent N: a mathematical algorithm loaded onto a stock computer. Beyond the luck of timing, it is difficult to understand why these patents are valid while the *Benson* algorithm was ruled as an invalid attempt to patent a mathematical algorithm.[70] It is clear, from these patents and a multitude of others, that there is no mathematical algorithm exception left. If an algorithm is too computationally intensive to be easily done by hand, or if the patent draftsman competently uses technical language, the applicant can evade the formerly recognized limitations on the patenting of mathematical algorithms.

H. The Logical Conclusion

Now that the "insignificant postsolution activity" rule has been thrown in the dustbin, what is the new line between patentable and unpatentable subject matter? Unfortunately, none exists. All human endeavors seem patentable when the only requirement to consider a process physical is something as ethereal as a cash payment or recorded share price. All traditionally non-patentable examples, such as a story written to paper, an inventive method of playing an F chord on the guitar, or a method of drawing a pictorial likeness of an object, involve both an inventive step and a step as physical (or more so) than many business method patents, meaning that any of them could take a form like Patent N.

Knight argues exactly this point. He explains that the CAFC's rulings (especially *In re Lowry*, regarding a data structure written to a hard disk) effectively overturned the "printed matter doctrine" that dictates that copyright, and not patent, protects writings.[71] Not one to merely publish in journals, his firm has submitted applications for patents on storylines and even new words;[72] as of this writing they are still pending. In *Ex parte Lundgren*, a

[70] One could perhaps argue that the algorithms can still be executed using paper and pencil rather than a computer, and therefore they are not truly pre-empted. However, the same could readily be said of the Benson algorithm—the ruling even instructs the reader on the steps. We should recognize that a law stating that mathematicians may use a computationally-intensive algorithm provided they do all calculations by hand is (in the words of Benson) to wholly pre-empt the algorithm "in practical effect." Gottschalk v. Benson, 409 U.S. 64, 72 (1972).

[71] Knight, *supra* note 4, at 865.

[72] *See* U.S. Patent App. 20050244804 (filed Nov. 28, 2003), *available at* http://appft1.uspto.gov/netacgi/nph-Parser?Sect1=PTO1&Sect2=HITOFF&d=PG01&p=1&u=%2Fnetahtml%2FPTO%2Fsrchnum.html&r=1&f=G&l=50&s1=%2220050244804%22.PGNR.&OS=DN/20050244804&RS=DN/20050244804 (describing the process of relaying a story having a unique plot (about a character who suffers a peculiar case of deja vu)); *See* U.S. Patent App. 20040249626 (filed Jun. 3, 2003), *available at* http://appft1.uspto.gov/netacgi/nph-Parser?Sect1=PTO1&Sect2=HITOFF&d=PG01&p=1&u=%2Fnetahtml%2FPTO%2Fsrchn

ruling regarding a method of paying managers, the USPTO's Board of Patent Appeals and Interferences (BPAI) failed to reject such applications: "Our determination is that there is currently no judicially recognized separate 'technological arts' test to determine patent eligible subject matter under §101. We decline to create one."[73]

While the case implies that BPAI sees no reason to exclude patents on storylines and neologisms, this determination may also offer a way out: could one write a technological arts test to exclude storylines and legal machinations but still allow share prices and virtual machines? This would be a novel rule. As the BPAI notes, there is no precedent for such a test in U.S. patent law. Further, one would be hard-pressed to write a section of the MPEP that would draw a clear distinction. Revisit the statement by Knuth, above: the only difference between the logical processing of a system of rules (tax law, social norms) and the processing of numeric data (share prices) is the symbols used.[74] Like many economics papers, patent draftsmen could easily rewrite their claims to include a wealth of Greek symbols, thus making a simple chain of logic look like a treatise on advanced mathematics. For example, millions of organizations use formal algebra based databases to keep track of customers, finances, and so forth. As such, one could describe database processes via either the relatively quotidian language of lists of names, dates, and sums, or via the language of abstract algebra. Since mathematics is invariant to changes of notation, patent law regarding mathematics should be as well.

To give another contentious example, consider patents on legal methods: business methods aimed at creative uses of laws such as tax regulations. To date, the USPTO has granted over a dozen such patents, however legal scholars and practitioners feel that such patents are bad policy.[75] Schwartz makes a number of policy arguments asserting why legal method patents are undesirable, all of which directly apply to software and business methods. He points out that "no attorney wants to pause before advising a client in order to run a patent search to make sure that no one 'owns' the advice she is about to give;"[76] the same certainly holds for a programmer sitting down to write a page of code, or a business manager considering how to arrange the furniture in her shop. Schwartz points out that (non-patent) law is a decentralized industry not focused on patented methods. Therefore, imposing patents will create

um.html&r=1&f=G&l=50&s1=%2220040249626%22.PGNR.&OS=DN/20040249626&RS=DN/20040249626 ("Method for modifying English language compositions to remove and replace objectionable sexist word forms" (in which HIM + HER becomes HIR, for example)).

[73] *Ex Parte* Lundgren, No. 2003-2088, 2005 Pat. App LEXIS 34, at *11 (B.P.A.I. Sept. 28, 2005).

[74] Knuth, *supra* note 30, at 1.

[75] Andrew A. Schwartz, *The Patent Office Meets The Poison Pill: Why Legal Methods Cannot Be Patented*, 20 HARV. J.L. & TECH.333, 335 (2007).

[76] *Id.* at 335.

immense transaction costs, a point fully explored in the next section. Schwartz explains that "the experience of the past two centuries has confirmed the theory that economic incentives are not needed to incentivize attorneys to innovate,"[77] and it takes absolutely no leap of logic to apply the same reasoning to business methods since the dawn of civilization, and software since the dawn of transistor arrays. Business methods, methods of saving costs and generating profits, especially do not need the implicit subsidy of a government-granted limited monopoly for development.

However, if our intuition and policy analysis indicate that storylines and legal methods should not be patented, and their lack of nontrivial physical manifestation does not preclude patentability, how is the law to exclude them? Schwartz proposes distinguishing "laws of man" from "laws of nature."[78] Here, a legal method and a storyline would constitute manipulations of laws of man, while software and business methods are manipulations of laws of nature. Unfortunately, such a distinction rehashes the debate regarding whether mathematics is a discovery from nature or an invention by man and therefore the same unresolved issues of categorization crop up again. The patent in *State Street* was over a method of calculating a price for an agglomeration of agglomerations of shares in legally-constructed corporations, what could be more man-made? Since the price is "accepted and relied upon by regulatory authorities," does that make it a method for compliance with law? Is the human nature revealed in a storyline a law of man or a law of nature? If you prick the characters, do they not bleed?

By eliminating the line that patented devices must be bona fide physical devices, it has become impossible to preclude legal methods, storylines, or other such structured writings from patentability. No new frontiers delimiting the scope of patentable subject matter have come forth to replace that erased by the CAFC, nor does it seem that a clear and enforceable candidate is forthcoming. Knuth's revenge rears its head again: information, being infinitely more malleable than physical materials and devices, can easily be recast to use whatever symbols and jargon the law requires, regardless of the underlying meaning. Simply put, once some categories of information are patentable, all categories are patentable. The rulings that allowed software to be patentable inexorably led to the patenting of legal methods and other such pure-information designs. But it is difficult to determine whether storyline and legal method patents are just amusing novelties, a brewing storm, or the dawn of a new age in tax loophole innovation. Better would be to look at the effects that patentability has had on software; since the *Alappat* ruling is over a decade old, there has been time for the effects of patents to be felt on this multibillion dollar industry.

[77] *Id.*

[78] *Id.*

II. THE ECONOMIC PERSPECTIVE

This section re-asks the same question as above: where should one draw the line between what is patentable and what is not? Instead of considering the statutory and judicial history, this section instead considers the structure of the software industry, and information-based 'industries' in general. It will show why allowing software and business methods to be patentable creates transaction costs that easily dwarf the benefits that such patent protection may provide. The key concept behind the discussion is that these pseudo-industries are massively decentralized, and patents do not efficiently promote progress in a decentralized industry. Unlike copyright, independent invention is not a valid defense against claims of patent infringement. If there are millions of potential independent inventors, then the waste and economic loss associated with restrictions on independent inventors becomes inevitable.

A. Empirical Evidence

The National Research Council (one of the National Academies) expressed concern that the expansion of patentable subject matter "has occurred without any oversight from the legislative branch,"[79] and that "the long-term effects of this trend [increasing software patentability] are as yet unclear, although the near-term consequences are worrisome."[80] The study points out that no research had been done on whether software patents will indeed foster innovation.[81]

So, now that software has been widely accepted as patentable for over a decade, have proxies for innovation such as the rate of software R&D investment shown an appreciable shift?

The answer is no.

Finding a jump in software innovation post-1994 would be difficult, since the computing world was so innovative without patents. The period when software was widely understood to not be patentable saw the invention of the Internet at large, the World Wide Web, email, instant messaging, word processors, relational databases, spreadsheets, graphical user interfaces, audio compression/decompression software, image manipulation software, some nifty games, and countless other options. One could even think of a few businesses that prospered before business methods were recognized as patentable in 1998. It is thus little surprise that no analyst has yet found that software patents are necessary (or even conclusively beneficial) for innovation in software.

Many analysts put a positive spin on the lack of findings. Lerner and Zhu optimistically point out that "little evidence can be found for harmful

---

[79] NATIONAL RESEARCH COUNCIL, THE DIGITAL DILEMMA: INTELLECTUAL PROPERTY IN THE INFORMATION AGE 228 (National Academy Press 2000).

[80] *Id.*

[81] *Id.*

effects."[82] Ronald Mann took the qualitative approach and interviewed venture capitalists and programmers, and found that none of them halted work because of software patents, but (as discussed further below) just ignored them.[83]

Mann and Sager looked at software startups, using a database of firms involved in seeking venture capital.[84] How did patents affect the odds that a startup would receive financing? "[W]e found no significance to the existence of pre-financing patents."[85] A suite of linear regressions found that the presence of patents toward the end of the study period, in 2005, was closely related to total investment, longevity of firm, rounds of financing, late-stage financing, and the exit status of the firm (closed down, went public, bought out, et cetera). For these regressions, p-values were 0.001 or smaller, indicating strong statistical significance.

But only one of these indicators was closely related to pre-financing patents: exit status (p=0.0157). However, the fact that four out of five regressions found no significant relation indicates that to take this one significant regression out of context would be to engage in a bit of "data snooping." The Bonferroni correction, common in biometrics, suggests that if five regressions were run, the p-values should be adjusted by a factor of five. The p-value should thus be interpreted not as 0.0157, but as 0.0785—not significant at customary levels. Taken as a whole, the results, that indicators of success are strongly correlated to late-stage patent rates but are at best weakly related to pre-financing patents, indicate that causality is not in the direction of patents fostering startup venture investment, but the other way around: successful firms eventually begin patenting, perhaps for defensive or other strategic purposes.

Robert Merges, another software patent advocate, makes this tepid endorsement: "I will venture a prediction: [p]atents will not cause any real and lasting problems . . . Something good may come of it."[86] Merges also produced an empirical paper along similar lines: it found that the software industry has not grown more concentrated since patents were introduced, and

[82] JOSH LERNER & FENG ZHU, WHAT IS THE IMPACT OF SOFTWARE PATENT SHIFTS? : EVIDENCE FROM LOTUS V. BORLAND 3 (NBER Working Paper No. 11168 2005), *available at* http://www.people.hbs.edu/fzhu/softwarepatents.pdf. Their analysis does indicate that organizations that patent more show greater profits. However, this does not show a social welfare improvement, but merely that those firms that are granted limited monopolies by the government make more money than those that are not. The search for overall benefits found only the lack of harmful effects cited in the main text. Similarly, Schankerman and Noel found that software patent rates are correlated to various measures of profitability, but made little effort to find whether there was social benefit to software patents.

[83] Ronald J. Mann, *Do Patents Facilitate Financing in the Software Industry?*, 83 TEX. L. REV. 961 (2005).

[84] Ronald J. Mann & Thomas Sager, *Patents, Venture Capital, and Software Start-Ups*, 36 RESEARCH POLICY 193, 195 (2007).

[85] *Id.* at 197.

[86] Robert P. Merges, *The Uninvited Guest: Patents on Wall Street*, FEDERAL RESERVE BANK OF ATLANTA ECONOMIC REVIEW 1, 12-13 (Fourth Quarter 2003).

that more experienced players are more likely to patent.[87] However, the empirics again offered little evidence to back up any claims that patents foster innovation, and Merges was again left with a tepid endorsement: "Whatever the effects of patents on the software industry, this paper concludes, they have not killed it."[88]

However, the purpose of patents is not to "fail to hinder the progress of Science and useful Arts," but to promote their progress. The Supreme Court, in its ruling in *Graham v. John Deere Co*.,[89] made it clear that the "promote the progress" clause is not a mere rhetorical flourish:

> The clause is both a grant of power and a limitation. This qualified authority, unlike the power often exercised in the sixteenth and seventeenth centuries by the English Crown, is limited to the promotion of advances in the "useful arts." It was written against the backdrop of the practices - eventually curtailed by the Statute of Monopolies - of the Crown in granting monopolies to court favorites in goods or businesses which had long before been enjoyed by the public. [. . .] The Congress in the exercise of the patent power may not overreach the restraints imposed by the stated constitutional purpose. Nor may it enlarge the patent monopoly without regard to the innovation, advancement or social benefit gained thereby. [. . .] This is the standard expressed in the Constitution and it may not be ignored.[90]

As the National Research Council noted above, patents were expanded to software and business methods without consideration of whether they promote the progress;[91] the analysis below will indicate that there are reasons to believe that patents in these fields will have more of a hindering effect than in other fields; even the most pro-patent empirical investigators were unable to find evidence of the promotion of progress; and software and business both flourished without patent protection. All of this indicates that the expansion of patentability to include these new subject matters may be unconstitutional.

B. Patent Efficiency In Massive Industries

The lack of findings is rather easy to explain. Imagine a study to consider the effect of business method patents on the business method industry—by

---

[87] ROBERT P. MERGES, PATENTS, ENTRY AND GROWTH IN THE SOFTWARE INDUSTRY 5 (Social Science Research Network 2006), *available at* http://papers.ssrn.com/sol3/Delivery.cfm/SSRN_ID926204_code69308.pdf?abstractid=926204&mirid=1.

[88] *Id.* at 2.

[89] Graham v. John Deere Co., 383 U.S. 1 (1966).

[90] *Id.* at 5-6.

[91] NATIONAL RESEARCH COUNCIL, *supra* note 89, at 228.

which I mean all businesses. It would be absolutely astonishing if the productivity of United States business could be proven to hinge on the few thousand business method patents granted to date. The story with software is similar. Define a potential innovator as someone who has sufficient human and physical capital that they could conceivably independently invent the innovation in question. For the typical pharmaceutical patent, the set of potential innovators includes only experienced chemists with affiliation to a research lab. On the other end of the spectrum, consider the famous Southwestern Bell Corp (SBC, now merged with AT&T) patents for a "Structured Document Browser,"[92] described in SBC's own words below. Here, the set of potential innovators includes anyone with a home computer and a basic knowledge of a web-design tool such as a word processor. Netcraft, an Internet data and consulting company, counted 106.9 million web sites in January of 2007.[93] By the definition here, the administrator of any one of these sites is a potential innovator.[94]

SBC actually pursued its claim against a broad range of companies with the misfortune of having a web site. For example, they wrote to one toy company:

> We recently observed several useful navigation features within the user interface or your site www.museumtour.com. For example your site includes several selectors or tabs that correspond to specific locations within your site documents. These selectors seem to reside in their own frame or part of the user interface. And, as such, the selectors are not lost when a different part of the document is displayed to the user . . . By separating the selectors from the content, Museum Tours has truly simplified site navigation and improved the shopping experience for its users. As you review the Structured Document Patent you will notice that the above-discussed features appear to infringe several issued claims in our patent.[95]

The salient feature of this letter is not that it describes what seems to be a rather obvious design—that issue will be discussed below. Rather, the key point is that SBC, a telephone company, was sending a software patent infringement letter to a toy company. It is unlikely that the owners of Museum Tour thought of themselves as a computer software firm, but by having a web

[92] U.S. Patent No. 5,933,841 (filed May 27, 1996); U.S. Patent No. 6,442,574 (filed Apr. 29. 1999).

[93] Netcraft, *January 2007 Web Server Survey*, Jan. 5, 2007, http://news.netcraft.com/archives/2007/01/05/january_2007_web_server_survey.html (one web site equals one domain name).

[94] *NTP, Inc. v. Research In Motion, Ltd.*, 392 F.3d 1336, 1370 (Fed. Cir. 2004).

[95] SBC Intellectual Property, *Re: US Patent No. 5,933,841 and US Patent No. 6,442,574*, Chilling Effects Clearinghouse, Jan. 10, 2003, http://www.chillingeffects.org/ecom/notice.cgi?NoticeID=537.

site, they effectively were.

The latest accounting from the Bureau of Economic Analysis divides the software market into three parts: retail, consultants, and in-house, which are evenly split in the U.S. economy. Of the $232.5 billion spent on software in 2002, 32.6 percent bought prepackaged programs, 36.4 percent custom-built ones, and 31.0 percent paid for software written in-house.[96] A report commissioned by the EU used a different method and different definitions that indicate that software sales represents an even smaller portion of the software economy: by their estimates, 16% of 2002 U.S. software sales went to proprietary software sales, 41% went to development and customization services, and 43% paid for internal development.[97] These statistics indicate that it is wrong to think of software patents as only affecting companies that vend software. The majority of software designers and authors are at companies where software is simply necessary for daily operation. Whereas the set of people who have the know-how, equipment, and inclination to work on a mechanical device may be limited to a few hundred organizations, the set of people who work full-time on writing new and potentially innovative software is distributed among millions of organizations. And the set of businesses that strive to use innovative business methods is simply the set of all businesses.

The size of the software and business method pseudo-industries derives directly from the elimination of the concept of insignificant postsolution activity. Information and the tools to manage information are truly ubiquitous, but the equipment needed to engage in significant postsolution activity, such as testing a chemical's properties or verifying that a machine actually runs, are available to a more limited range of people. Therefore, when inventions with no significant postsolution activity are allowed, the number of potential innovators grows exponentially. This also provides justification for the mathematical algorithm exception to patent law. The set of people who use mathematics is simply the set of all humans, so all of the problems regarding applying patent law to pseudo-industries that cover the entire economy also apply to the set of all mathematics users. Thus, even if one believes that there are no ethical objections to the patenting of mathematical results, there are pragmatic reasons for treating it exceptionally. Information processing is simply different from the processing of tangible ingredients.

[96] U.S. DEPARTMENT OF COMMERCE, BUREAU OF ECONOMIC ANALYSIS, RECOGNITION OF BUSINESS AND GOVERNMENT EXPENDITURE FOR SOFTWARE AS INVESTMENT: METHODOLOGY AND QUANTITATIVE IMPACTS, *1959-98*, Feb. 2, 2007, www.bea.gov/bea/papers/software.pdf (updated with 2002 data at www.bea.gov/bea/papers/table11.xls).

[97] Study on the: Economic impact of open source software on innovation and the competitiveness of the Information and Communication Technologies (ICT) sector in the EU, final report, 20 November 2006, 124.

C. Scaling

Let us say that a producer has a product that embodies a patent. On the one hand, additional research could provide benefit to a larger number of people because patents provide more benefit in a massive market. On the other hand, the fact that the potential market is millions instead of thousands means that it is much easier to find customers, and it is possible to profit on smaller per-unit margins. So there is less need for government intervention in the market because patents provide less benefit in a massive market. One could write a model and numbers to tip the scale either way. We can say more about how the number of potential innovators (rather than potential customers) will affect the value of a patent. The likelihood that someone will stumble upon and reveal an innovation is larger when there are more agents searching. If, in a no-patent world, there are already millions of people in a field searching for a new innovation, then patents are less likely to add value. Some argue that "patent races" induce faster innovation. The idea is that if there are multiple parties that could all potentially achieve an invention, and the first will be granted a monopoly on the result, then all parties will invest more in research — and do so more quickly. The drawback to the patent race, however, is that the investments made by the losers in the race are basically unproductive lost costs. In the economic literature, different papers present different results, depending on how the patent race is modeled. Dasgupta and Stiglitz find that "competition in R&D increases the level of innovation, possibly beyond the socially optimal level."[98] Conversely, Sah and Stiglitz state that "the number of firms has no effect on the pace of innovation in a market economy."[99] I take the economic literature to give weak support to either claims that patent races raise or decrease the benefit from patents.

Now consider the transaction costs of patents. Transaction costs include the cost of negotiating licenses with the monopolist, the costs of patent searches, and the cost of legal disputes. For a clear patent, the cost of negotiating licensing could be small: the monopolist may develop a standardized take-it-or-leave-it price list, and those who know they will need a license will quickly get through the necessary paperwork. Of course, the paperwork rises with the number of actors, but we can generally expect that the effect of a larger number of innovators is positive but small. This is especially the case if there is a bottleneck where users can be collectively licensed. For example, Apple paid Creative Technology $100 million for the right to implement its method of sorting songs on the iPod,[100] and so the iPod's millions of users implicitly

[98] Partha Dasgupta & Joseph E. Stiglitz, *Uncertainty, Industrial Structure, and the Speed of R&D*, 11 BELL J. of ECONOMICS 1, 1 (1980).

[99] Raaj Kumar Sah & Joseph E. Stiglitz, *The Invariance of Market Innovation to the Number of Firms*, 18 RAND J. of ECONOMICS 98, 99 (1987).

[100] *Apple Agrees to Pay $100 Million to Resolve IPod Patent Litigation*, L.A. TIMES, Aug. 24, 2006, at C2; *Apple Pays $100 million U.S. to Settle Patent Suit*, TORONTO STAR, Aug. 24, 2006, at C03; Nick Wingfield, *Apple, Creative Technology Set Pact Ending iPod-*

licensed the technology from Creative.

There is also the cost of determining that there is a patent and finding the patent owner (patent search). To truly eliminate the risk of patent infringement, all industry members must do regular patent searches. If the cost of a patent search were fixed, twice as many participants would mean twice as much spent on searching. This is especially true because subtle differences in operations can make for entirely different patent situations, so it is difficult for multiple parties to pool patent searches. One would be hard-pressed to find an attorney who would work for such rates, but say that a minimal patent search costs about $10,000. Then if every potential infringer of a software patent—all 106 million web sites—took U.S. patent law seriously, the transaction costs would total about $1 trillion. As it is, the world's web economy progresses by mostly ignoring U.S. patent law.

On top of this, add those businesses that engage in business methods—that is, all businesses. When deciding how to set price lists, a manager is a potential innovator who must do a patent search and take action on the results to ensure that the price list is not infringing any patents. Again, this would impose astronomical costs if taken seriously; the economy moves by simply ignoring the requirements of patent law.

D. Patent Trolls

One final transaction cost is the possibility of litigation. The cost of litigation, when it occurs, easily dwarfs the cost of a patent search. In an ideal world litigation never happens, and parties are all sufficiently aware of the law that they can all stay out of the courtroom.[101] The worst case would be a situation where the members of one subset of the industry hold patents, and the members of another subset are entirely ignorant of those patents. In such a case, the likelihood of wasteful infringement suits rises significantly over a situation where everyone is equally informed of what is patented. This is exactly the case in software and business methods. Because the set of potential innovators is so large, the likelihood of independent invention is high, and the likelihood that an independent inventor has done a patent search is low. SBC had little difficulty finding a number of web sites like that of the toy company that infringed its patent.

It is an easy syllogism: there are 100 million potential independent inventors; independent invention is not a valid defense against claims of patent infringement; therefore opportunistic infringement claims are basically inevitable. In this context, we see the rise of the patent troll. One prevalent definition of the term[102] is that a patent troll is a company that holds a patent not to produce a product but to extort royalties from others. However, non-producing patent holders are common. Until an inventor finds an investor to

Tied Dispute, WALL ST. J. (Eastern Edition) Aug. 24, 2006, at A2.

[101] *See*, *e.g.*, RICHARD A. POSNER, ECONOMIC ANALYSIS of LAW § 21.5 (3rd ed. 1986).

[102] Attributed to former Intel Assistant General Counsel Peter Detkin.

provide capital, she remains a non-producing patent holder. If she shows the blueprint to an investor, and he uses it without permission, then she has every reason to sue. Although such suits are wasteful by themselves, one can argue that they are beneficial in the long term because they support the system that allows inventors and capital-holders to fruitfully interact. But it is much more difficult to argue that a lawsuit against an independent inventor in a different field entirely provides any social benefit. At best, the suit is simply court costs wasted on rent-seeking. At worst, such a suit could create the stifling impression that one may not engage in any but the most obvious activity without first consulting a lawyer. Thus, a more useful definition is that a patent troll is one who unfairly takes advantage of informational asymmetries by suing independent inventors who are ignorant of the field of patents in which the troll works. Informational asymmetries often lead to outcomes that allow the agent with more information to extract rents from agents with less information, and which are socially suboptimal.[103]

In a symmetric situation, both the patent holder and potential licensees are aware of the patent. Therefore the potential licensees can plan accordingly, by either negotiating a patent license before embarking upon any major investments, or by passing on to other prospects. In an asymmetric situation, the potential licensee invests its assets into its business without knowing about the patent. This produces an asymmetric bargaining position where the patent holder may sue for infringement damages up to the full value of the company. To give one example, Kodak demanded a billion dollars from Sun Microsystems due to a patent infringement regarding its Java compiler and virtual machine, which Sun gives away for free.[104]

Looking at the list of companies that the media have dubbed patent trolls, including Acacia, BTG, DataTreasury, Eolas, Forgent, and NTP, all of them work in software and business methods, and this is no coincidence. There are no asymmetries of information in pharmaceuticals or mechanical engineering, because everyone is aware of the patent process, and for any given field there are a limited number of players (tens or hundreds, not tens of millions) so the costs of patent search are limited. Because the number of parties are small, the likelihood that another party independently invented a patented item is relatively small. Conversely, for software or business methods, the question is not whether someone ignorant of the patent has also implemented it, but how to find that party and what to bill them.

The problem with patent trolls has clearly taken hold in the media[105] and

[103] *See, e.g.*, HAL VARIAN, MICROECONOMIC ANALYSIS, Ch. 25 (3rd ed.1992).

[104] John Oates, *Kodak Wins Sun Java Patents Case, Wants $1bn*, THE REGISTER, Oct. 4, 2004, http://www.theregister.co.uk/2004/10/04/kodak_wins_java/; Ashlee Vance, *Sun Settles Java Spat with Kodak for $92 Million*, THE REGISTER, Oct. 7, 2004, http://www.theregister.co.uk/2004/10/07/kodak_sun_settle.

[105] *See, e.g.,* William M. Bulkeley, *Freedom Wireless to Get $55 Million in Patent Accord*, WALL ST. J., July 22, 2006, at A9; Alan Cane, *Trolls Control the Rickety-Rackety*

even in Congress,[106] but is it all merely anecdotal? Even if it is, the anecdotes by themselves are impressive. In 2006 alone, we saw payments of $612.5 million to NTP from RIM (the makers of the Blackberry),[107] $92.3 million to Acacia from various companies throughout all industries (in first 3Qs of 2006, plus $27.6m in 2005),[108] and $8 million to Fogent from a coalition of licensees.[109] So in 2006 to date, these few opportunistic lawsuits took in $712.8 million. This includes only opportunistic threats against companies that independently invented the software in question, and it should be no surprise at this point that these cases are entirely about software. To give a sense of scale, Google made $733 million in net income in the third quarter of 2006.[110]

Since Microsoft is perceived as the deepest pockets in the traditional software industry, it is frequently the target for software infringement suits. Settlements and rulings against the giant include $521 million to Eolas (settled for an undisclosed amount),[111] $440 million to Intertrust,[112] $60 million to SPX,[113] and $60 million to Burst.com.[114] We may want to include in this the

Bridge of Intellectual Property and Until a Way Can be Found to Stop Them, Business Will Continue to Live in Fear of Them and Consumers Will Continue to Foot the Resulting Bill, FINANCIAL TIMES (London), Sept. 20, 2006, § FT Report – Digital Business, at 2; Julie Creswell, *A Wall Street Rush to Patent Profit-Making Methods*, N.Y. TIMES, Aug. 11, 2006, at C7; Tim Harper, America's *Patent Trolls: Are They Out of Control?*, TORONTO STAR, Feb. 3, 2006, at A01;*'Patent Trolls' and Market Dominance*, WASH. TIMES, Oct. 4, 2006, at A18; Bruce Sewell, *Troll Call*, WALL St. J., Mar. 6, 2006, at A14; Richard Waters, *Invention Shop or Troll Factory?*, FINANCIAL TIMES (London), Apr. 26, 2006, § Business Life, at 2;David Wilson, *Patent Trolling a Thorn in the Side of Hi-tech Giants*, S. CHINA MORNING POST, May 23, 2006, § Technology, at 2.

[106] Based on interviews with members of the Senate Subcommittee on Intellectual Property.

[107] *Settlement Reached in BlackBerry Patent Case,* MSN (Associated Press), Mar. 3, 2006, http://www.msnbc.msn.com/id/6448213/did/11659304?GT1=7935.

[108] Acacia Technology Group Fact Sheet, http://www.acaciaresearch.com/pr/AcaciaFactSheet.pdf (last visited Dec. 28, 2006). (SOURCE DOES NOT HAVE CITED INFO)

[109] Michael Kanellos, *Forgent Settles JPEG Patent Cases*, CNET NEWS, Nov. 1, 2006, http://news.com.com/2100-1014_3-6131574.html.

[110] Google Inc., Quaterly Report (Form 10-Q), at 4 (Sept. 30, 2006), *available at* http://www.sec.gov/Archives/edgar/data/1288776/000119312506228163/d10q.htm.

[111] Michelle Kessler, *The 521-million-dollar Man*, USA TODAY, Sept. 30, 2003, at 3B; *Microsoft Settles a Dispute Over a Feature in Its Browser*, N.Y. TIMES (Bloomberg News), Aug. 31, 2007, at C4.

[112] Intertrust Press Release, http://www.intertrust.com/main/ip/settlement.html (last visited Nov. 11, 2007).

[113] Keith Regan, *Microsoft Settles Patent Lawsuit for $60M*, TECHNEWSWORLD, Dec. 29, 2003, http://www.technewsworld.com/story/32479.html.

[114] Peter, Borrows, *Underdog Or Patent Troll? How Burst.com Went from Making Software to Suing Tech Giants*, BUSINESSWEEK ONLINE, Apr. 24, 2006, http://www.businessweek.com/magazine/content/06_17/b3981070.htm

$900 million Microsoft paid to Sun for patent licensing, but it is unclear whether the claims were over independent invention or actual imitation of a competitor's products.[115]

Adding up the settlements we find a variety of companies, some in traditional software and many elsewhere, paying billions of dollars for the right to use software they conceived and wrote without outside assistance—and those are just the headlines. I exclude settlements such as DataTreasury's 2006 settlements with BankOne and JP Morgan, in which the amount paid to license independently written software was surely in the hundreds of millions but the exact amount was not disclosed to the public.

Furthermore, there is no reliable way to document the constant stream of letters and settlements that fly under the radar, though there are a few rules of thumb presented by experts that may help us get a ballpark figure. Chip Lutton of Apple Computer[116] offers a rule of 25: for every litigated case, plaintiffs file twenty-five, and for each of those twenty-five they file, plaintiffs send twenty-five notice letters claiming a patent infringement. When a company receives a notice letter it must meet compliance requirements which costs between $30,000 and $50,000;[117] when we estimate the number of notice letters from the rule of thumb, we get approximately $18.7 to $31.2 million in legal fees spent on handling notice letters and other low-level compliance matters for every patent lawsuit filed. With 55 software patent suits filed per week[118] , these costs quickly grow rather large. Although it is near impossible to get a solid estimate of software patent compliance costs, the above rough guides indicate that an estimate in the hundreds of millions would not be unreasonable. It may be that opportunistic lawsuits against independent inventors are mere anecdotes in the sense that they do not comprise a sufficient data set for rigorous time-series analysis, but when the anecdotes total billions of dollars, we must pay attention to them, nonetheless.

A decade from now, can we expect that there will be more or fewer opportunistic infringement suits? There are two competing forces that will determine whether we will see more or fewer opportunistic infringement claims in the future. On the one hand is the patent reform. It is slow and may not work, because it is simply impossible to halt patent trolls—by either definition of the term—while allowing the rest of the patent world to continue

[115] Stephen Shankland, *Sun Settles with Microsoft, Announces Layoffs*, CNET NEWS, Apr. 2, 2004, http://news.com.com/Sun+settles+with+Microsoft,+announces+layoffs/2100-1014_3-5183848.html.

[116] Patent Quality Improvement: Hearing Before the Subcomm. on Courts, the Internet, and Intellectual Property of the H. Comm. on The Judiciary, 109th Cong. 122 (2005) (Testimony by Richard J. Lutton, Jr., Chief Patent Counsel, Apple, on behalf of the Business Software Alliance (BSA)). Thanks to Brian Kahin for pointing out this testimony.

[117] Daniel Ravicher, Presentation at the Brookings Institution: Software and Law: Is Regulation Fostering or Inhibiting Innovation? (Dec. 7, 2005).

[118] *Id.*

unabated. As discussed below, any type of patent reform in Congress has stalled since software patents became an important issue. On the other hand, the key bottleneck in filing an opportunistic lawsuit is capital, and we may expect that that will be easier to obtain it. A fully-prosecuted infringement case can cost millions of dollars, and runs some risk of losing it all. Each of the half-billion dollar settlements above bolsters the business case for those who want to enter the opportunistic infringement lawsuit industry. That is, successful patent trolls breed other patent trolls, so we can expect that the problem will only get worse in the near future. Patents have become significant in massively decentralized industries only during the past decade, and a fully prosecuted patent suit takes several years. Then, it is impressive and noteworthy that we have already seen so many opportunistic infringement suits. But we may expect that in the future, more people will comprehend the syllogism that given a hundred million potential independent inventors, and independent invention not a defense, profiting is easy.

E. Contributory Infringement

If a car manufacturer infringes on a patent on its windshield wipers, every owner of the car is a contributory infringer, and is theoretically equally liable under the law.[119] Thus, one could argue that the car industry creates patent liability that is as massively decentralized as the software industry. There are three key differences between the circumstances above and those associated with software that further elucidate the many ways infringement can occur in a massively decentralized industry. First, unlike the theoretical contributory infringement suit above, plaintiffs have claimed software-patent infringement lawsuits against a very broad range of non-software organizations. As an example, Acacia Media Technologies brought suit against Johns Hopkins University, the University of Virginia, the University of Wyoming, and a number of other schools claiming that some online components of their classes violate Acacia's patents.[120] Second, under the logic of *In re Alappat*, the act of loading software onto a general purpose computer creates a "new machine," so loading software onto a computer is direct infringement, unlike the contributory infringement of driving a car with unlicensed windshield wipers. Acacia did not sue universities for contributory infringement but for direct infringement. Third, and most importantly, there is very little unmodified software running in the world's server rooms. Without fail, one needs to reconfigure off-the-shelf software to work with the local situation, perhaps even rewriting the program's internal algorithms, which is why a third of software spending is in-house. Such behavior creates another opportunity for direct infringement. The toy company may create its web site using an off-the-shelf web design program, but the company employee designed the page in a

[119] 35 U.S.C. § 271(a); 35 U.S.C. § 271(c).

[120] Corey Murray, *Schools Targeted in Streaming Video Patent Claim*, ESCHOOL NEWS, Mar. 3, 2004, http://www.eschoolnews.com/news/PFshowstory.cfm?ArticleID=4937.

manner that infringed the SBC patent. Thus, it is wrong to describe the world as consisting of a few centralized software authors and a wide array of downstream consumers: the downstream users actively write new code and modify the existing code, blurring any distinction between the author and user.

F. Is It All Just Obviousness?

Some contend that the entire problem with software and business method patents—the reason that only these industries and none other have a serious problem with opportunistic lawsuits—is that patent examiners for these industries allow obvious patents that would not be allowed in traditional industries where the patent record and USPTO experience is much more extensive. Of course, "patent quality" and the line between obviousness and non-obviousness will always be a problem, and I will not dispute that many seemingly obvious patents have been granted. Fixing the obviousness problem would do nothing, however, to alleviate the problems with applying patents to a massive industry.

First, let us say that there were only 200 patents in the software industry instead of 200,000. Every web site would still need to verify that it was in compliance with those patents. One can think of the cost of patent clearance as an affine function: there is a fixed cost (say, $10,000) for retaining a lawyer to search one patent, plus an additional cost for searching each additional patent (say, a nickel). In such a case, of the large number of actors in the industry, about a hundred million, and not the number of patents, potentially makes the search costs astronomical.

The USPTO takes several years to grant a patent, meaning that the above discussion about independent invention is invalid only if a patent is nonobvious both at the date of application and several years later after the grant date. SBC's above-mentioned patent is often mentioned as the perfect example of an obvious patent, but on the date of application (May 1996), graphical web browsers were still new and in limited use. But as millions of people began using the medium, what had been non-obvious and obscure quickly became obvious to millions.

Some would contend that subjects like the aforementioned "Method for efficient modular polynomial division in finite fields f(2m)" is unlikely to have a massively decentralized audience like the "Structured Document Browser" does. That is, the problem of massively decentralized software production could be alleviated via a technical arts requirement. As discussed above, it would be basically impossible to write such a test. The Structured Document Browser patents are already couched in the technical language of GUI rendering of SGML DTDs, and if the draftsman had to, she or he could likely express it all in terms of the manipulation of byte arrays in certain hexadecimal registers.

It is natural to presume that as one raises the standards for what is obvious, the likelihood of independent invention falls. In an ideal world, we would define nonobvious to mean that the likelihood of independent invention is low,

and would assign patent examiners to evaluate this likelihood. In such a world (which is very different from the one in which we live), examiners would assign different standards to different fields. Eliminating the risk of independent invention with a few hundred integrated circuit (IC) manufacturers involves a much lower standard than what would be required to eliminate the risk of independent invention among 100 million web sites. So the first step in eliminating obvious patents in software would be to establish differential standards for patentability in different fields. But recall the fiction that loading software on a computer creates a "new machine;" to maintain this fiction, the USPTO has no separate software category. So as the next step in implementing higher standards for software patents, the USPTO must categorize software separately from ICs—but the courts base the validity of software patents by claiming that it is impossible to make such a distinction. While this would not be the first self-contradictory law in U.S. history, we must not encourage such laws. Our options are thus to set an incredibly high standard for integrated circuits for the sake of better software patents, continue to tolerate lawsuits against independent inventors in software for the sake of allowing IC manufacturers to continue as they do now, or to acknowledge that software and ICs are distinct—which would be to overturn the rulings that Patent N is a bona fide physical device like any IC.

G. Disclosure

Campbell-Kelly points out that patents also serve the purpose of disclosing innovative ideas to the public that the inventor would have otherwise kept secret.[121] With an astronomical number of potential innovators at a comparable level to the patent-holder, this means that the benefits to disclosure may also be great. But the disclosure story, while very sensible in theory, has a number of important problems in the real world. The first is that firms in all industries tend not to search patent databases for technological instruction. Arora et al. state that "patent disclosures appeared to have no measurable impact on information flows from other firms, and therefore no measurable effect on R&D productivity."[122] Arundel finds that "a consistent result in survey research on the use of patent databases is that they are among the least important external information sources available to firms." [123] On top of this base, there are further problems with disclosure among software and business methods. In Northern *Telecom v. Datapoint*, the CAFC ruled that writing source code—in fact, authoring anything more detailed than the broad flowchart describing the overall logic of the design, is a "mere clerical

[121] Martin Campbell-Kelly, *Not All Bad: An Historical Perspective on Software Patents*, 11 MICH. TELECOMM. TECH. L. REV. 191, 226 (2005).

[122] Ashish Arora et al., *R&D and the Patent Premium* 17 (Nat'l Bureau of Econ. Research, Working Paper No. 9431, 2003).

[123] Anthony Arundel, *Patents in the Knowledge-Based Economy*, 37 BELEIDSTUDIES TECHNOLOGY ECONOMIE 67-88 (2001).

function."[124] Therefore, the disclosure requirements of a software patent are minimal. So if a patent does not need to reveal any but a basic flowchart, what does it reveal?

On one end of the software spectrum are patents for user interface designs such as the SBC patent, whose workings are evident from inspection, so the patent provides no disclosure. Any patent on a non-software design, such as a business method or storyline, would also fall into this class, and would therefore provide no informational value beyond the product itself. On the other end of the spectrum, one finds the algorithm for LZW (Lempel-Ziv-Welch) encoding, which Campbell-Kelly cites as a success story for patents.[125] As Campbell-Kelly notes, however, Terry Welch had published the algorithm in a peer-reviewed journal shortly after applying for the patent, but years before receiving it.[126] If Mr. Welch had truly wanted to keep the LZW algorithm a trade secret should patent protection be unavailable, then he would have waited for the ink on his patent application to dry before publishing the algorithm.

To generalize from the anecdote of Mr. Welch, computer science at the level of advanced algorithms for data processing is still more concerned about journal publications and academic recognition than it is about patents, and one is hard-pressed to find a programmer who searches the patent literature for disclosure of ideas. Mann interviewed software technologists and found that the apathy toward learning from patents revealed by the general surveys is evident in software as well: "None of the startup firms to which I spoke suggested a practice of doing prior art searches before beginning development of their products."[127]

Thus, although disclosure is a positive benefit from patents in some fields, an overall perception among technologists that patent searches are not an effective manner of learning new techniques, the weak interpretation of the enablement requirement, the self-evident nature of many software and business method designs, the difficulty of searching patents relative to the ease of searching journals and online code bases, the academic nature of the more arcane aspects of computer science, and the speed of the industry all collude to make patents on software and other information-processing technologies virtually useless for disclosure purposes.[128]

---

[124] Northern Telecom., Inc. v. Datapoint Corp., 908 F.2d 931 (Fed. Cir. 1990).

[125] Campbell-Kelly, *supra* note 121, at 227;

[126] Terry A. Welch, *A Technique for High Performance Data Compression*, COMPUTER, June 1984, at 15-16.

[127] Mann, *supra* note 83, at 1004 (italics in original).

[128] Returning to the scaling question, the fact that there is no evidence of computer scientists or business method users searching patents for research purposes means that the benefits from a patent do not spread to other members of the industry. So if the social benefit to a software patent's disclosure in a world with a thousand potential innovators is $100,000, then the benefit in an industry of a hundred million potential innovators is still

III. CONCLUSION

To summarize the example of the software industry, empirical analysis shows that the millions of players basically ignore patents. However, a few lucky losers are sued for independently written software, and must spend up to millions of dollars defending against claims of infringement. Such lawsuits are not a result of obviousness or details of patent procedure. If a patent holder has a preeminently valid patent and there are 100 million others working in the field, the patent holder need only search to find someone who is unknowingly infringing. As the current crop of litigation shows how easy it is to find unknowing infringers, one can expect that the next generation of litigants will have an easier time funding their own searches for infringers.

The problems with patents in a massively decentralized industry stem from the elimination of the "insignificant postsolution activity" stipulation on patentable subject matter. Virtually every company in the country has a computer on hand, and many have full-time employees actively writing code. Yet only a handful of companies have employees actively synthesizing new drugs. This is closely related to the fact that information processing is a general requirement for anyone with information, and that information processing tools are considered stock equipment.

Thus, the recommendation of this paper is not an innovation, but a regression. A great many of the problems with patents that fill the newspapers and vex businessmen can be solved by reinstating the distinction from *Diehr* and its predecessors that indicate a device is patentable only if it is based on steps that are simultaneously novel and non-trivially physical. By respecting the caveats about postsolution activity in the Freeman-Walter-Abele test, the printed matter doctrine would be re-established, because the process of printing to paper, a ribbon, or a hard drive is not physically innovative, regardless of the written data's inventiveness. The rule that a patent be physically innovative in no way hampers the traditional process patents that manipulate materials into new materials. A process such as a novel means of folding a sheet or processing flour can not be reduced to a design step that makes no mention of the materials and an insignificant postsolution activity step executed with little regard to the algorithm.

Some will argue that processing a data structure and writing it to a hard drive are intrinsically linked—that this line is too difficult to distinguish in practice. Yet the USPTO managed to distinguish this type of physical process from more traditional physical processes for decades. The distinction certainly created more disputes and borderline cases than the CAFC's position, but that is because the CAFC ended any debate by simply eliminating subject matter restrictions entirely.

Patent applicants work hard on blurring all distinctions and, with enough flowery jargon, stock computers and even the process of putting pen to paper

\$100,000. But as above, the risk of opportunistic and socially destructive litigation increases significantly as the number of potential innovators expands.

can look like innovations. But the patent examiner has the ability to limit the scope of any claims. For example, the examiner may add an annotation limiting the claim to the invention when executed using physical steps beyond those executed by stock equipment. This would be more than sufficient to distinguish, say, a novel computer chip fabrication from the same logic executed in software, or a new portable email device from a new means of using stock networks to transmit emails.

Further, the concerns about whether patent examiners will be fooled are overblown by a simple observation bias problem. Law review articles and key court cases always treat the edges of patentability, but the great majority of software patent applications are clearly of the form of Patent N: an algorithm loaded onto an obvious stock computing device. Respecting the declaration that "insignificant postsolution activity will not transform an unpatentable principle into a patentable process" would easily eliminate the great majority of such patents without any need for fine judgment calls.

A. Politics

How can limits on patentable subject matter be restored? As of this writing, there are a number of venues in government for debate over the course of patent law. In the Executive Branch, the USPTO introduced a draft strategic plan that proposed reforms such as a multi-tiered system that one patent attorney described as "a deconstruction of our unitary patent system."[129]

In Congress, both the House and Senate have introduced bills, but given the current condition of patents, it is unlikely that any beneficial reform will come from them, for two reasons.[130] First, the CAFC rulings that software companies and pharmaceuticals are beholden to the same patent law mean that building consensus on any given patent reform is suddenly twice as difficult. As a very broad rule, pharma companies generally seek stronger patent protection, while software companies tend to seek more restrictions on patentability; building consensus from two groups pushing in opposite directions is especially difficult.[131] The second problem is the collective action problem. As explained above, there are about 100 million potential targets for a patent infringement suit regarding a web page, and as many potential targets regarding a patent on a business method as there are businesses. However, a relatively minuscule number of individuals and organizations actually hold patents. The few who stand to profit significantly from their patent business are

[129] J. Matthew Buchanan, *Friday Food for Thought: The Changing Scope of 'Patent Reform'*, PROMOTE THE PROGRESS, Sept. 22, 2006, http://promotetheprogress.com/archives/2006/09/friday_food_for_thought_the_changing_scope_of_patent_reform.html.

[130] This section is primarily based on personal interviews with various Congressional staffers working in some capacity on the Senate Subcommittee on Intellectual Property.

[131] Alan Sipress, *Patently at Odds: Drug and Tech Sectors Battle With Reform High on Agenda*, WASHINGTONPOST.COM, Apr. 18, 2007, *available at:* 2007 WLNR 7515235.

much more likely to lobby Congress than the diffuse masses.[132]

The other option for restoring limits to patentable subject matter is the courts—and since the CAFC has shown a specialist's enthusiasm for allowing patents on all subject matter, it falls to the generalist Supreme Court to consider what patent limitations best promote progress. As discussed above, members of the Supreme Court have shown interest in defining patentable subject matter, via the dissent in the rejection of certiorari in *LabCorp v. Metabolite*. Given that technicalities precluded reaching a verdict in that case, one would expect that the Court is currently seeking a new test case in which to discuss patentable subject matter. Potential test cases abound that raise the patentability of designs consisting of novel software loaded on a stock computer.

All the pieces fit together neatly. Patent law, the only IP regime where independent invention is not a valid defense,[133] is a terrible match to massively decentralized pseudo-industries like software authorship or business methods, where there are so many players that independent invention is basically inevitable. The reason that these pseudo-industries are so decentralized is that they handle pure information with a trivial after-step to apply the information to human affairs. There is a history of court rulings stating that pure information processing is not patentable, even when a patent draftsman adds "insignificant postsolution activity" to apply the information to real-world affairs. Thus, this judicial line distinguishing the patentable from the unpatentable exactly matches the ideal economic line that divides traditional industries that prospered with patents from the massively decentralized information-based industries that have prospered without patents.

---

[132] *See* MANCUR OLSON, THE LOGIC OF COLLECTIVE ACTION: PUBLIC GOODS AND THE THEORY OF GROUPS 44 (Harvard University Press 1971) .

[133] Fortunately for all involved, there is a type of intellectual property protection that covers information and yet does allow independent invention as a valid defense: copyright. As I discuss elsewhere, an appropriately interpreted copyright law provides exactly the level of protection needed by software algorithms. *See* BEN KLEMENS, MATH YOU CAN'T USE: PATENTS, COPYRIGHT, AND SOFTWARE 131 (Brookings Institution Press 2006).