9 Abstract Patents and Software

9.1 Introduction

In Chapter 7, we noticed that patents on software and especially patents on business methods (which are largely software patents) stood out as being particularly problematic. These patents had high rates of litigation and high rates of claim construction review on appeal. This chapter explores whether there really is something particularly awry with software patents and, if so, what it is.

We argue that there is, in fact, something crucially different about software: software is an *abstract* technology. This is a problem because at least since the 18th century, patent law has had difficulty dealing with patents that claimed abstract ideas or principles. In Chapter 3, we noted that abstract patent claims can violate the rule of first possession, allowing patent holders to lay claim to arguably broad ranges of technology that they have not invented. Such patents often have unclear boundaries and give rise to opportunistic litigation. Here we explore how software can be considered "abstract" and how this affects patent notice.

Although not all software patents contain abstract claims, the technology facilitates abstract claiming. In addition, court decisions in the 1990s have undermined legal doctrines that restricted abstract claims in software patents in particular. Software also seems to be an area with large numbers of relatively obvious patents. For these reasons, it is not surprising that a substantial share of current patent litigation involves software patents.

9.1.1 Are software patents different?

A number of commentators have argued that patents on software or business methods are no different from patents in other technology fields. The evidence from chapter 7 and other evidence we present below casts serious doubt on this assertion.

The evidence supporting the assertion is quite limited. Allison and Tiller (2003) and Hunter (2003) argue that business method patents are of comparable "quality" to other patents because they have similar number of claims, citations, etc. as other patents. But this conception of patent "quality," however, is quite limited and does not relate at all to how these patents function as a property right.

Others admit that there may be some difficulties with software patents, but they argue that these difficulties are temporary. For example, Martin Campbell-Kelly (2005), a computer science professor, argues that "software patents are not radically different from those of other technologies; the patent system has adapted to the particular demands of new technologies over time, and the software patent system is already making such adaptations." He concludes

> It is ten years since software patents have been issued in large numbers. The anxieties expressed in the early 1990s about the effect patents would have on the software industry have not been realized. History shows us that software patents are not so different from other patents in the information technology industries, and that the patent system is capable of adjusting to the particularities of individual industries. For example, early in the last century chemical processes were thought to be unpatentable, but the system soon adapted to a new reality and now it is difficult to imagine this issue was once controversial. With software and business method patents the US Patent and Trademark Office is instituting changes that will make the system work better. For example, it has increased the number and quality of software patent examiners, and in time

the databases and searching mechanisms for prior art will improve.[1]

There are several things wrong with this argument. First, Campbell-Kelly's history is wrong. Chemical processes have always been patentable and this has never been controversial in the US. The very first patent granted in 1793 was for a chemical process to make potash and many famous chemical processes were patented during the nineteenth and early twentieth century (e.g., the Solvay process, the Haber process, etc.).

Moreover, in US history, no other technology has experienced anything like the broad industry opposition to software patents that arose beginning during the 1960s. Major computer companies opposed patents on software in their input to a report by a presidential commission in 1966 and in *amici* briefs to the Supreme Court in *Gottschalk v. Benson* in 1972.[2] Major software firms opposed software patents through the mid-1990s (for example in USPTO hearings in 1994). Perhaps more surprising, software inventors themselves have mostly been opposed to patents on software. Surveys of software developers in 1992 and 1996 reported that most were opposed to patents (Oz 1998). Although other countries have witnessed general opposition to patents in the past (for example, some European countries abandoned the patent system during the 19th century) and although some countries have opposed patents on certain technologies in the past (for example, some countries permitted patents on manufacturing processes for pharmaceuticals, but not patents on the chemical structures themselves), such broad opposition from *within* the affected industry and among the affected inventors seems to be unprecedented in the U.S.

Second, it is hardly clear that the "anxieties" expressed about software patents have not been realized. The principal "anxiety" Campbell-Kelly identifies concerns the development of so-called patent thickets. Citing an article written by Richard Stallman and Simon Garfinkle in 1992, Campbell-Kelly writes, "Opponents of software patents argue that patent 'thickets' will necessarily impede the flow of new software products." Campbell-Kelly is not alone in arguing that rumors of the death of the software industry from patent thickets have been greatly exaggerated. Merges (2006) agrees that "the predictions of the software patent doubters in the early 1990s have been effectively refuted so far." He argues that the US software industry has not become dominated by large firms nor has it shown signs that the entry of small firms has diminished.[3] Mann (2005) makes a similar argument based on interviews with small software firms.

Now, credible evidence shows that software publishers have flourished so far despite the growth of software patents. Some preliminary studies suggest that although there is evidence of some detrimental effects of patents within the software industry, the effect of patents within the industry has

1 A similar argument is made by David Kaefer, Microsoft's director of business development for the IP and Licensing group. In a recent interview (Kerner 2005) Kaefer argued that issues surrounding the patentability of intangibles is nothing new and that there was a debate over patents on electrically-powered consumer appliances during the 1920s. According to Kaefer, it was argued that "electricity is a force of nature and no company should have control over nature." This seems to be a rather strange reading of history since electrical devices were patented since the early 19th century and some of the most famous 19th century patents were electrical (e.g., telegraph and telephone) and were held valid by the Supreme Court.

2 Many companies have changed their positions over time. IBM initially opposed software patents, then supported them enthusiastically during the 1990s, but recently it has grown concerned about overly generous protection. In recent years, Microsoft has been a stronger defender of software patents than most IT firms, but even Microsoft supports certain substantive reforms that limit patent protection.

3 Merges's evidence is based on an analysis of the software publishing industry as a whole. Usually studies of industry concentration and entry examine individual market segments separately. Indeed, Cockburn and MacGarvie (2006), using the same data set, look at entry into individual market segments and find that patents do have a negative effect on entry rates.

Table 1. Litigation characteristics by patent technology

	Probability patent in suit	Relative frequency of claim construction appeal	Percent of lawsuits (2002)
ALL	2.0%	1.00	
Chemical	1.1%	0.84	13%
Complex (excluding chemical)	2.0%	0.89	34%
Other	2.2%	1.11	
Software	**4.6%**	**2.18**	**26%**
business methods	13.7%	6.67	4%
Biotechnology	3.2%	2.37	3%

Note: Probability of suit adjusted for truncation bias and under-reporting.

not been serious to date. Cockburn and MacGarvie (2006) find that patents deter entry in the software industry, all else equal, but that this deterrent effect is muted because entrant firms can improve their odds of success by getting patents, too. Noel and Schankerman (2006) find that patent thickets reduce the market value of software firms, but this effect is muted, too, because software firms can counteract it by obtaining patents themselves.

But the arguments of Campbell-Kelly, Merges and Mann seem to be largely directed at a carefully chosen straw man. The general concern is over software *patents*, not the software *industry* per se. This distinction is important because almost all software patents are obtained by firms *outside* the software industry. Bessen and Hunt (2007) find that the software publishing industry only obtains 5% of all software patents granted; most are obtained by firms in electronics, telecommunications and computer industries. Software, of course, is a widely applied, general purpose technology and only about one third of all computer programmers and system analysts are employed in software publishing and software services industries.

Patents have little negative effect within the software publishing industry to date because there *are no* substantial patent thickets within the industry. Cockburn and MacGarvie look carefully at software industry market segments from 1994-2004 and find that in most segments, 80-95% of the incumbent firms have *no* patents related to that segment. This does not mean, however, that software patents do not contribute to patent thickets in *other* industries. Indeed, many of the industries that obtain the lion's share of software patents, such as semiconductor and computer industries, have been identified by multiple researchers as having patent thickets.[4] Nor does this mean that software patents may not create a thicket in the software industry in the *future*, as software firms are accelerating their patenting. Concerns about thickets in the software industry may simply be premature. The software industry might very well follow the path of the semiconductor and computer industries.

More generally, Campbell-Kelly, Merges and others focus heavily on this one "anxiety"about patent thickets when, in fact, a wide variety of other concerns have been raised about software patents since the 1960s, including some during the early 1990s (see, for example, the comments by software

4 And these might adversely effect firms in the software industry, however, the effect of these patents on software firms is likely not as strong as the effect of a patent thicket within the software industry itself.

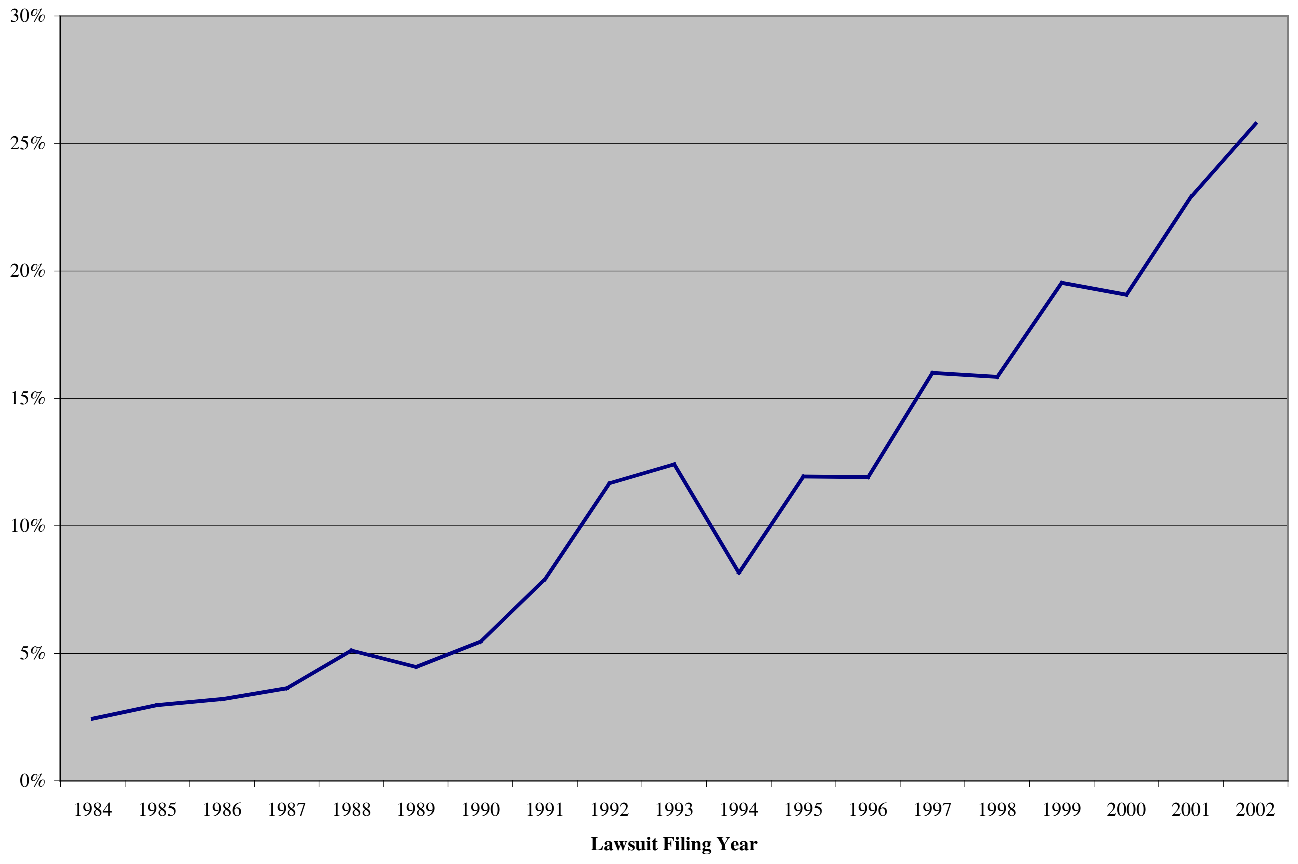

Figure 1. Percent of patent lawsuits involving software patents

firms at the Patent Office hearings held in San Jose in 1994).[5] Many of these anxieties concerned the ability of the patent system to properly handle software patents, including concerns about how well patent boundaries could be defined.

The empirical evidence suggests, in fact, that some of these concerns might be quite valid. Software patents play a major role in the rise of litigation that we have identified as central to the deteriorating performance of the patent system. Table 1 shows several characteristics of patent by technology type. The first two columns, repeated from Chapter 7, show the probability that a patent is in a lawsuit and the relative frequency with which claim construction is reviewed on appeal. Recall, the second column measures the uncertainty of claim construction. The last column shows the (non-exclusive) percent of lawsuits filed in 2002 that involve a patent of the given technology type. Over a quarter of all patent lawsuits involve software patents. Moreover, that percentage share has been growing rapidly. Figure 1 shows the share of lawsuits involving software patents over time.

Clearly, software patents and business method patents are different from most other patents in their litigation rates and claim construction problems. And a very major part of the poor performance of the patent system can be attributed to lawsuits involving these patents. It is therefore hardly surprising that major IT firms have become very vocal in their advocacy of patent reform. Although many of these firms support the use of software patents, one can hardly argue that the use of software patents has been without "anxieties."

5 Bill Gates was among those who raised concerns during the early 1990s about the effects of patents on software innovation (Warshofsky 1994).

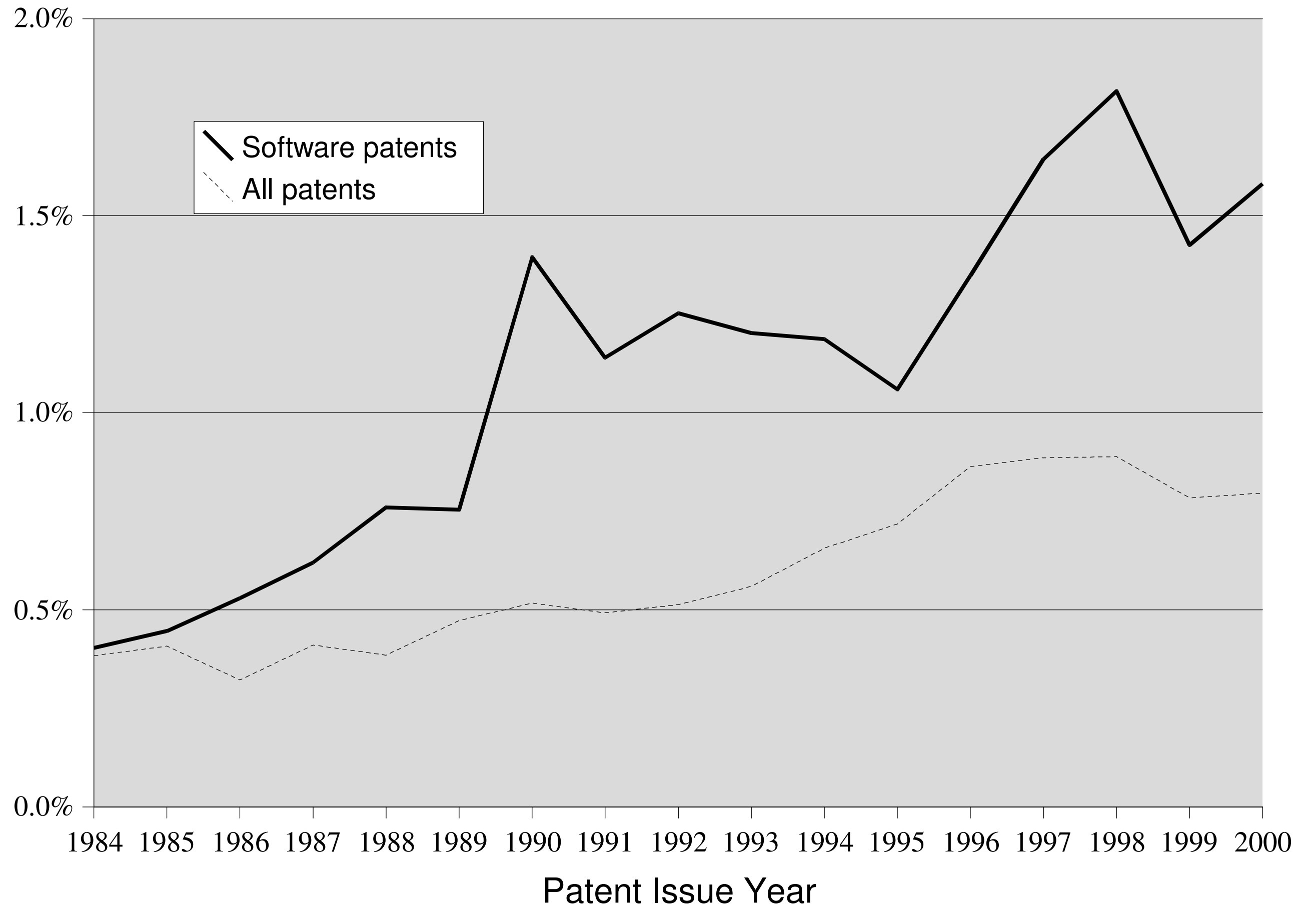

Figure 2: Probability patent is in lawsuit within 4 years of issue

Third, Campbell-Kelly argues that whatever problems exist, the patent system is already adapting to solve them. Indeed, the Patent Office issued new guidelines in 1996 for examining software patents, the Patent Office has begun hiring large numbers of electrical engineers and computer scientists, and a huge body of patent prior art has developed in software (over 150,000 patents). Significant numbers of software patents have been issued since the 1980s, so if the problems of software patents just require adaptation to this new technology, then we would expect the adaptation to be well underway, as Campbell-Kelly suggests.

However, the empirical evidence suggests that the problems of software patents are not diminishing; if anything, they are getting worse. Figure 2 shows the probability that a patent will be in one or more lawsuits within four years of the patent's issue date (adjusted for under-reporting[6]). Software patents issued in more recent years are much *more* likely to be litigated, not less. The adaptations made by by the patent system do not seem to be have produced a net positive effect on litigation rates. This suggests that something more persistent is at work here than the adaptation of patent examination procedures to a new technology.

6 The source of these data are Derwent's Litalert database, which is known to report only about two thirds of the actual patent lawsuits (Lanjouw and Schankerman 2004, Bessen and Meurer 2005). Following Bessen and Meurer, we correct for under-reporting by dividing the observed litigation rates by .64.

9.2 Software patents/abstract patents

9.2.1 Some problematic software patents

We believe that on average software patents suffer notice problems more acutely than patents drawn from most other areas of technology. There also seem to be large numbers of obvious software patents, aggravating these notice problems. Taken together, these factors probably explain why software patents impose higher litigation costs than other types of patents.

The following disputes illustrate some of the reasons why software patents have unpredictable boundaries.

4,528,643, "System for Reproducing Information in Material Objects at a Point of Sale Location" (Interactive Gift Express, Inc. v. Compuserve Inc., 256 F.3d 1323)

This is the widely-litigated Freeny patent discussed in Chapter 1. On appeal, five different claim terms were disputed.[7] Freeny's actual invention was a kiosk or vending machine to be used in retail locations for producing digital music tapes or other digital reproductions. One of the disputed claim terms was "point of sale location." The district court judge interpreted this limitation to mean that the patent did not cover transactions that occurred in private homes, e.g., as with much consumer digital e-commerce. In industry jargon, this term refers to the location within a retail store where items are checked out and transactions take place. This jargon was first used during the 1970s when electronic terminals began replacing cash registers. On appeal, however, the Federal Circuit interpreted this term more broadly to cover transactions that occurred within private homes. This meant that the patent could cover a wide range of e-commerce applications far beyond Freeny's original invention, and the owner of this patent (E-Data) has asserted it against companies ranging from the New York Times to Compuserve to McGraw-Hill.

Another disputed claim term was "material object." The district court interpreted this to mean that the technology had to produce a digital reproduction in something separate from the computer itself; the computer's internal hard drive did not count as a "material object" for the purposes of this patent. In this case, the Federal Circuit supported the district court's narrower interpretation. Nevertheless, even with the narrow interpretation of this term, this patent still appeared to cover a wide range of activities far beyond the original invention, activities that merely shared a functional similarity to the original invention.

The practice of using vague, expansive terms such as "material object" is by no means limited to

7 The first claim of the patent reads, "1. A method for reproducing information in material objects utilizing information manufacturing machines located at point of sale locations, comprising the steps of:

providing from a source remotely located with respect to the information manufacturing machine the information to be reproduced to the information manufacturing machine, each information being uniquely identified by a catalog code;

providing a request reproduction code including a catalog code uniquely identifying the information to be reproduced to the information manufacturing machine requesting to reproduce certain information identified by the catalog code in a material object;

providing an authorization code at the information manufacturing machine authorizing the reproduction of the information identified by the catalog code included in the request reproduction code; and

receiving the request reproduction code and the authorization code at the information manufacturing machine and reproducing in a material object the information identified by the catalog code included in the request reproduction code in response to the authorization code authorizing such reproduction."

software technology. But software may be particularly susceptible to the use of abstract terms because many of the standard terms of art are themselves abstract ideas that are meant to apply to a wide variety of possible applications. That is, software is itself an abstract technology.

The abstract use of standard terms is illustrated by the following patent.

4,751,669, "Videotex Frame Processing"
(Wang Lab., Inc. v. America Online, Inc., 197 F.3d 1377)

Videotex was a 1970s vintage, online service that was a precursor to the modern Internet. Individual users could log on to a centralized server and display a variety of different types of information on a local terminal or (later) personal computer. Compuserve and Prodigy were popular videotex systems in the US.

Wang Labs entered this market in the early 1980s with a system that used Wang PCs connected to servers over telephone lines. The software running on the personal computers used an interface modeled after the graphical user interface developed at Xerox PARC during the 1970s and early 1980s that was used in the Smalltalk and Xerox Star products. In particular, the Wang software displayed information in "frames," overlapping rectangles each containing different information. Frames are, of course, quite familiar to users of windowed operating systems such as the Apple Macintosh operating systems and Microsoft Windows. In contrast to the Xerox Star and the later systems, however, Wang offered a "poor man's" implementation: the low resolution monitor on the Wang PC was not capable of displaying the high resolution bit-mapped graphics used on these other systems, but only displayed text characters and crude graphic characters.

Nevertheless, Wang obtained a patent in 1988 that claimed, among other things, the general use of frames to display different kinds of information retrieved from servers.[8] This patent putatively covered a broad range of possible technologies that might use a graphical user interface far beyond Wang's actual product. The Internet browser was one of these later-developed uses. Accordingly, Wang sued Netscape and America Online in 1997. This immediately caused great concern among Internet companies.[9]

The judges, however, interpreted the claim term "frame" narrowly. Since Wang's actual system, as described in detail in the patent, only used a character-based display, the court concluded that the term "frame" as used in the patent only pertained to character-based displays, despite the origin of the term and its industry usage. Because the Netscape browser used a higher resolution bit-mapped display, it was found not to infringe.

8 One relevant claim reads, "38. Apparatus for locally processing frames of information received from central videotex suppliers, different frames being encoded in accordance with different protocols, comprising

means connected to locally store the information frames,

means connected to locally display the frames,

means connected to decode the locally stored frames as they are displayed, and

means connected to tag each stored frame with a header indicating one of said different protocols as having been used for encoding the frame,

the means connected to decode being arranged to decode each frame in accordance with the protocol indicated by the header on the frame."
This claim is written in "means-plus-function" form but the court considered the term "frames" generally as well as the in the specialized context of a means-plus-function claim.

9 See for example, "Netscape enlists help to fight Wang lawsuit," Network World, May 4, 1998 by Gittlen, Sandra.

This meant that a wide range of Internet and other applications were not subject to this patent. Although industry participants were relieved by this decision, they could hardly have expected it, given the industry usage of "frame." The unpredictable nature of claim construction meant that the boundaries of the patent were surely uncertain for many years. Wang was certainly not unreasonable in attempting to assert what it thought was a legitimately broad patent, even though Wang did not itself invent the Internet browser.

Indeed, because of the abstract nature of software, claims made in one field may read on technologies in very different fields. The following example illustrates this situation.

5,758,257, "System Method for Scheduling Broadcast of and Access to Video Programs and Other Data Using Customer Profiles"
(Pinpoint, Inc. v. Amazon.com, Inc., 369 F. Supp. 2D 995)

A common set of programming techniques match objects in one list to objects in another. How broadly should claims to matching programs be interpreted?

In 2003, Pinpoint, a company whose only business is patent licensing, sued Amazon, the online bookseller, because Amazon's website provided personalized book and movie recommendations to customers. Pinpoint had obtained a patent that claimed a system for recommending TV programs to viewers based on past choices.[10] But the patent claimed a more general procedure, often called "collaborative filtering," for matching "customer profiles" (information on customer preferences, e.g., customer likes romance novels) to "content profiles" (information on content, e.g., book is a romance novel). Pinpoint felt that its TV technology was claimed broadly enough to cover Amazon's book recommendation technology.

Amazon argued, among other things,[11] that the method Pinpoint actually described used a numeric comparison to do the matching, even though the patent claim did not limit the comparison to a comparison of numeric quantities. Amazon claimed that its technology did matching instead by comparing categorical descriptions, e.g., romance novels, mystery novels, etc. The first judge (there were multiple hearings) disagreed with this interpretation. The second judge, instead, agreed with Amazon, arguing that Pinpoint's patent did not cover Amazon's technology because Pinpoint's patent only covered matching "based on mathematically comparing mathematically expressed customer preferences with mathematically expressed program contents." This was deemed not to cover matching based on categorical descriptions.

10 One relevant claim reads: "43. A method of scheduling customer access to data from a plurality of data sources, comprising the steps of:

creating at least one customer profile for each eligible recipient of said data, said customer profile including a profile of data previously accessed by said customer;

creating content profiles for each data source of said data, said content profiles reflecting the customer profiles of those customers who have previously accessed said data from each data source;

relating said at least one customer profile with the content profiles for the data available from each data source to the customer;

determining a subset of data having content profiles which are determined in said relating step to most closely match said at least one customer profile; and

presenting said subset of data to said customer for selection."

11 Another issue in the claim interpretation was whether the preamble to the claims that used the word "scheduling" limited that matching to temporal matching. Patent law has complicated rules for determining whether words in the preamble limit the claims.

This broad interpretation, which might have covered a wide range of matching and collaborative filtering applications, was thus narrowed. The second judge in this case (Richard Posner) seems to have made a pragmatic decision that this patent did not deserve such broad coverage of applications very different problems from the original invention.

Arguably, the broad interpretation matched the *plain meaning* of the claim. No computer programmer reading this claim prior to Judge Posner's decision would distinguish between matching routines based on mathematical expressions and matching routines based on categorical descriptions. Indeed, any programmer would see this is as a distinction without a difference: in a computer, matching based on categories *is* a mathematical routine; the software simply translates verbal descriptions into mathematical expressions.[12]

We don't point this out to cast stones at Judge Posner or to question the mathematical competence of judges generally. That would be an undeserved cheap shot. First, Judge Posner's decision was based on more than this distinction. More important, this example is really a simple illustration of a basic problem: defining the boundaries of software patents precisely can be difficult, perhaps impossible. As we shall see below, even the best computer scientists and mathematicians in the world may not be able to unambiguously define the boundaries of software patent claims.

* * *

We argue that the patents in these disputes share a common characteristic: they feature abstract claims. Of course abstract patent claims are not limited to software patents. We now turn to look at abstract patents, explaining what they are, what is wrong with them and how the law restricts them (or not).

9.2.2 The problem of abstract patent claims

It is often stated that patents are not granted for abstract ideas or principles, but only for practical inventions that might be based on such ideas or principles. The patent statute itself contains no such explicit restriction, the courts, however, have interpreted various provisions to imply such a limitation.

As we noted in Chapter 4, courts have resisted patents on abstract "principles of manufacture" since the 18th century in Britain and a similar restriction has been part of patent law in the U.S. at least since the first half of the 19th century. For example, faced with a patent that claimed "the art of cutting ice by means of any power, other than human power," Justice Story in 1840 argued,

> "Such a claim is utterly unmaintainable in point of law. It is a claim for an art or principle in the abstract, and not for any particular method or machinery, by which ice is to be cut. No man can have a right to cut ice by all means or methods, or by all or any sort of apparatus, although he is not the inventor of any or all of such means, methods, or apparatus. (Wyeth v. Stone, 30 F. Cas. 723)"

This language evokes the fox hunting case, *Pierson v. Post,* and affirms the principle that possession is a key feature of property rights. Abstract claims like this tend to be problematic because they stray too far from the invention the inventor actually possessed. It is doubtful the inventor knew "any or all of such means, methods, or apparatus" to cut ice. The distinguishing feature of an abstract patent claim is not that it covers a broad range of technologies, although that is often the case, but rather that it claims technologies unknown to the inventor.

12 To be sure, these are integer expressions as opposed to real number expressions, but integer expressions are a subset of real number expressions.

The patents in the lawsuits above fit this description. Freeny did not anticipate or know about Internet e-commerce when he filed his patent application, nevertheless, e-Data subsequently and plausibly read the words in his patent to cover e-commerce. Wang did not conceive of the general Internet browser, but Wang interpreted the language in its patent to cover the operation of such browsers after they emerged.

The process of interpreting patent claims is one of mapping the words in a patent to a range of technologies, much as a surveyor maps the words in a deed to demarcations on the ground. With abstract patent claims, however, the words cover unknown territory, claiming technologies that are unknown at the time the patent is filed and which may change over time, especially in fast moving fields of technology.

There are two inter-related problems with such abstract claims. First, these claims reward patentees for inventions they do not invent. This means that the actual, future inventors face *reduced* incentives because they have to obtain a license from the patentee to develop or to commercialize their inventions. Clearly this counters the social benefit of the patent system.

Second, it may be difficult to determine the boundaries of such claims and thus it may be difficult to provide notice, to conduct clearance searches, or to even determine the content of the prior art. The problem of mapping words to technology is difficult and it is made more difficult if the claims are not tethered to a specific device or to a specific physical or chemical process. Patent lawyers use the phrase “the embodiments of the invention” to describe the specific devices and processes disclosed in the patent document. Courts often interpret the meaning of the words in a claim in light of the specific embodiments of the invention. The mere idea of, say, cutting ice, cannot provide the same important information about patent boundaries that concrete embodiments provide.

The Freeny, Wang, and Pinpoint patents illustrate several notice problems caused by abstract claims in software patents. First, the words in an abstract claim map to an uncertain set of technologies when they are not limited to distinct embodiments. This was the problem with “frame” in the Wang claim and with “matching” in the Pinpoint claim. Sometimes, the progress of technology will render this mapping increasingly uncertain over time.

Second, software patents may be particularly prone to strategic use of vague language by applicants to gain undeserved scope. This was the problem with claim terms "point of sale location" and "material object" and "information manufacturing machine." Although clever lawyers can use vague language with any technology, abstract technologies particularly lend themselves to such abuses because they are inherently described in abstract terms.

Third, notice problems can also arise with abstract claims even when the words themselves seem quite clear. This may seem counterintuitive, but it often happens when a patent claims all techniques for achieving a desired result, as in the case of the claim to “the art of cutting ice by means of any power, other than human power.” Judge Story, recognizing that this patent claimed far more than the technology actually invented, narrowed the patent scope to correspond more closely to the actual embodiment of the invention. Arguably, the judges in the Wang and Pinpoint cases also narrowed patent scope for similar reasons, that is, they recognized that such broad claims are *inequitable* and they undermine the purpose of the patent system by penalizing the actual inventors of later-developed technology. Sometimes, it is quite clear upfront that claims will be interpreted narrowly—the patent statute provides for such narrowing when the claim only describes a general means for achieving a function. Nevertheless, in a broad range of cases, significant uncertainty remains as to whether abstract claims would be upheld, and if so how they would be interpreted if challenged in court. This means that these claims have uncertain boundaries, giving rise to opportunistic disputes and litigation. Even though judges may often (but not always!) take a pragmatic approach and interpret such claims narrowly—as they seem to do often with software patents—the uncertainty about boundaries makes clearance difficult and subjects inventors to risk of inadvertent infringement.

9.3 Why software patents are different

But these problems of abstract patent claims clearly apply to a broad range of technologies in addition to software. Why should software have a particular problem with abstract patents? We argue that the abstractness of software technology inherently makes it more difficult to place limits on abstract claims in software patents. To make things worse, the courts have undermined the legal doctrines that might be used to contain this problem. Finally, the nature of software technology combined with the courts' relaxation of patentability standards means that there are large numbers of trivial, seemingly obvious software patents that make clearance infeasible.

9.3.1 Can we know the boundaries of software inventions?

Patent law depends on comparisons between technologies. Determining whether a technology is novel depends on a comparison to prior technologies. Determining whether a technology infringes depends on a comparison to a patented technology.[13] These comparisons determine the effective boundaries of the patent.

In the examples above, this determination was difficult or uncertain. Partly this was because clever lawyers drafted patent claims using vague, expansive terms such as "material object." In other cases, lawyers argued for interpretations of claim terms that affected the range of covered technologies. Such lawyering exists for all patented technologies. Nevertheless, patent law assumes that once the words are mapped to a specific set of technologies, one can readily determine which technologies are equivalent and which are distinct.

However, for software, this assumption is not always true. Computer algorithms may be equivalent, but computer scientists may not *know* that they are equivalent. In many cases, it has taken years for them to discover that different techniques are equivalent. For example, it has been shown that the "traveling salesman" problem, which is used for routing delivery trucks among other things, is more or less equivalent to the "map coloring" problem and a whole range of other problems. This means that an algorithm for solving the traveling salesman problem is also, if worded broadly enough, an algorithm for doing map coloring. In other cases, computer scientists suspect that algorithms are equivalent, but they are unable to prove the equivalence.

Other observers have also noted the disparate representations of computer algorithms and its possible significance for patent law (Newell 1986, de Laat 2000, Klemens 2006). Our point here is that these different representations of the same technology create critically difficult problems for the notice function of the patent system.

Consider, for example, Karmarkar's patent on an algorithm for linear programming (4,744,028). Linear programming is a method used to optimize the operation of petroleum refineries, to schedule airline flights and to route long distance telephone calls. A method was developed to solve these

13 In patent law, these determinations (and others) are typically made by a hypothetical "person having ordinary skill in the art" (PHOSITA). PHOSITA has to look at two descriptions of technologies and determine whether the underlying technologies are equivalent or distinct.

problems on computers during the 1940s, but this method becomes very slow as the size of the problem (e.g., the number of telephone switches) grows. Narendra Karmarkar, a researcher at Bell Labs, developed a new, faster algorithm for linear programming in 1984, which AT&T touted as a "breakthrough."[14] AT&T filed a patent on this algorithm in 1984 and obtained a patent in 1988. AT&T also developed a product (KORBX), bundling the software with a high performance computer. Although AT&T did make a small number of sales and also apparently licensed the patent, this effort was not a significant commercial success.

This patent is sometimes cited as an example of what a software patent should be: a highly specific, non-trivial contribution to practical knowledge. However, serious questions exist as to the boundaries of this patent, whether its claims are truly novel, and whether Karmarkar actually "possessed" all the technologies claimed. One problem is that Karmarkar's algorithm seemed similar to technologies developed during the 1960s. In 1986, computer scientists demonstrated, in fact, that Karmarkar's algorithm is equivalent to a class of techniques that was known and applied to linear problems during the 1960s (Gill et al. 1986). Moreover, after this equivalence was demonstrated, computer scientists began applying algorithms based on these older techniques to linear programming problems and some of these algorithms appeared to work better than the Karmarkar-AT&T approach (Marsten et al. 1990).

The reason that the older technology had not been widely applied to linear programming problems during the 1960s was that a critical complementary technology had not been developed, namely techniques for efficiently performing calculations for sparsely-populated matrices (Marsten et al. 1990, p. 112). The new wave of linear programming applications following Karmarkar—including AT&T's offering itself—depended on these new matrix calculation techniques. Indeed, without these techniques, Karmarkar's algorithm by itself was not particularly efficient compared to the linear programming techniques of the 1940s. But these matrix techniques were neither claimed nor mentioned in Karmarkar's patent.

Given these facts, consider the difficulty of determining the boundaries of this patent. Would anyone have seen Karmarkar's algorithm as novel in light of the techniques used in the 1960s? Certainly not after 1986, when their equivalence was proved. But even in 1984, computer scientists might well have had doubts, yet they would have been unable to make a certain comparison (see for instance the account of one operations researcher, Saltzman 1994). Similarly, would AT&T have been able to assert its patent successfully against people who used linear programming techniques based on the techniques used in the 1960s? Apparently AT&T was able to obtain a cross-license from IBM, which had used these older techniques.

The abstractness of the patented algorithm means that these determinations cannot be made with certainty. Patent law assumes that two technologies can unambiguously be determined to be equivalent or distinct; this determines the patent boundaries. Yet for software, this assumption simply does not hold. Although this assumption works for most other technologies, it does not work well for software algorithms. And if computer scientists cannot make these determinations with any certainty, how can we expect judges and juries to do so?

Of course, not all software patents cover algorithms. Some are quite specific and limited in what they claim. Yet others, as in the examples above, use abstract terms that are either inherently subject to expansive interpretations or they use aggressive patent drafting to extend the reach of their claims. But in these cases as well as with algorithms like Karmarkar's, the abstractness of the technology means

14 "Breakthrough in Problem Solving," N.Y. Times, Nov. 19, 1984, at A1, col. 3.

that, taken as a whole, software patents are more prone to unclear notice than are many other technologies.

9.3.2 Patent Law and Software

Patent law has developed several different doctrines to grapple with the treatment of abstract patent claims (these doctrines are discussed further in the Advanced Topic appendix). However, since 1990, the Court of Appeals for the Federal Circuit has significantly undercut these doctrines as applied to software inventions.

In general, patent law restricts abstract patents with three doctrines:

1. Subject matter. Various decisions have excluded certain abstract subject matter such as mathematics and natural laws. Some decisions require a physical or chemical transformation or structure, another way to exclude claims that are too abstract.

2. Enablement. The disclosure principles of patent law require that the patent should provide sufficient information to enable a person skilled in the art to make and use all the inventions claimed. For example, in a famous case, Samuel F. B. Morse included the following claim to his patent on the telegraph:

> "Eighth. I do not propose to limit myself to the specific machinery, or parts of machinery, described in the foregoing specifications and claims; the essence of my invention being the use of the motive power of the electric or galvanic current, which I call electro-magnetism, however developed, for making or printing intelligible characters, letters, or signs, at any distances, being a new application of that power, of which I claim to be the first inventor or discoverer."

The Supreme Court feared that future inventions were likely covered by Morse's eighth claim and therefore the Court invalidated it. Although the Morse opinion is sometimes read as concerning patentable subject matter, the Court argued that the patent failed to document the processes or machinery these future inventions would use, nor could it, because Morse did not know them.

3. Limits on functional claims. The patent statute directly authorizes and regulates one style of abstract claim language called "means-plus-function" language. When a claim term recites only a general means of performing a function (as opposed to actually describing the specific means), the term is limited to the specific structures described in the patent document and their equivalents. This provision facilitates patent notice; interested observers interpret the otherwise abstract term by locating concrete technology described in the patent.

Patent law appears to take a belt-plus-suspenders approach to dealing with abstract patent claims. Perhaps this is because it is difficult to pin down exactly what sort of abstract claiming is impermissible.

How well do these legal doctrines limit abstract claims in software patents? Unfortunately, the Federal Circuit has made several decisions specific to software patents that undermine the performance of these doctrines:

The Morse case teaches that a high level functional description of electronic communications technology was not adequate to serve as an enabling disclosure for his eighth claim. However, beginning in 1990, a series of decisions by the Federal Circuit allowed similar high level functional descriptions of software inventions to satisfy this requirement.[15] Software patents do not, in general,

15 Northern Telecom, Inc. v. Datapoint Corp., 908 F.2d 931; Fonar Corp. v. General Elec., Co. 107 F.3d 1543. Ironically,

need to include the computer code nor detailed flowcharts nor any other detailed description of specific operation. In practice, this has meant that the disclosure requirement is rarely grounds to invalidate software patents, despite widespread use of claims that are every bit as abstract as Morse's eighth claim.[16]

The evolution of subject matter restrictions for software is a bit more complicated (see Samuelson 1990 for an overview). In its 1972 decision in *Gottschalk v. Benson*, the Supreme Court recognized that patents on mathematical algorithms might run afoul of the problems of Morse and the common law restrictions against patenting laws of mathematics. Despite popular misconceptions, the Court did not prohibit patents on software. Instead, the decision had the effect of subjecting patents involving software to an additional two-step test designed to limit abstract claims: the first step was to determine whether the patent recited a mathematical algorithm; if so, the second step asked whether the patent would preempt all uses of the algorithm. Such claims would not be allowed.

The application of these rules was controversial and confused, in part because the appellate courts saw things rather differently than the Supreme Court. The law evolved until 1994 when, in *In re Alappat*, the Federal Circuit effectively discarded this subject matter test and replaced it with the simple criterion that inventions involving software are patentable as long as they are "useful, concrete and tangible." By this, the court apparently means that an invention is not abstract as long as it has some practical application (the words "concrete" and "tangible" are apparently superfluous).[17] Although this is one meaning of the word "abstract," it is not the meaning favored by the Supreme Court in *Benson,*[18] which we have identified as causing notice problems. In that case, as well as in Morse's eighth claim and the ice-cutting claim, abstract language allowed the patent to claim all means of achieving a result or all technologies with similar functional form. The tests based on *Benson* recognized that the abstract language of software algorithms might be particularly problematic in this regard. With *Alappat* those concerns are ignored. Indeed, in 1998 in *State Street Bank & Trust Co. v. Signature Financial Group, Inc.*, the Federal Circuit concluded that a software patent that claimed all means of calculating accounting ratios for certain kinds of mutual fund groups did, in fact, meet the "useful, concrete and tangible" test. In effect, the use of any subject matter test to limit abstract patent claims has been eliminated for software.

Until recently, the notice enhancing restrictions on means-plus-function language have not had

the Freeny patent makes claims that seem surprisingly similar to Morse's eighth claim: instead of printing at a distance we have the production of digital media in material objects. Freeny adds some additional requirements about exchanging codes, but Morse limits the claim to techniques using electromagnetism.

16 One recent exception is LizardTech, Inc. v. Earth Res. Mapping, Inc., 35 Fed. Appx. 918 (2005).

17 In State Street Bank & Trust Co. v. Signature Financial Group, Inc., the Federal Circuit elaborated that "...certain types of mathematical subject matter, standing alone, represent nothing more than abstract ideas until reduced to some type of practical application, i.e., 'a useful, concrete and tangible result.' Alappat , 33 F.3d at 1544, 31 USPQ2d at 1557. Unpatentable mathematical algorithms are identifiable by showing they are merely abstract ideas constituting disembodied concepts or truths that are not 'useful.' From a practical standpoint, this means that to be patentable an algorithm must be applied in a 'useful' way. In Alappat , we held that data, transformed by a machine through a series of mathematical calculations to produce a smooth waveform display on a rasterizer monitor, constituted a practical application of an abstract idea (a mathematical algorithm, formula, or calculation), because it produced 'a useful, concrete and tangible result'-the smooth waveform."

18 The *Benson* Court notes that it is considering a subject matter issue: "The question is whether the method described and claimed is a 'process' within the meaning of the Patent Act." Then the court uses language similar to ours: "Here the 'process' claim is so abstract and sweeping as to cover both known and unknown uses of the BCD to pure binary conversion. The end use may (1) vary from the operation of a train to verification of drivers' licenses to researching the law books for precedents and (2) be performed through any existing machinery or future-devised machinery or without any apparatus." *Gottschalk v Benson*, 409 U.S. 63 (1972)

much effect on software patent claims. However, beginning in 1999, the Federal Circuit has required that patents deemed to use such wording must disclose a specific algorithm.[19] This is a small step in the right direction. Its value may be limited though because of the broad and uncertain range of mathematical equivalents to algorithms, and because patent applicants can always avoid means-plus-function language and find another way craft abstract claims.

Judging from our reading of Federal Circuit cases, the net effect of these changes is that there are few limits to abstract software patent claims. Software patents that make broad, abstract claims are, in general, held to be valid.[20] For software patents, the court has made specific exceptions to general principles of patent law.

This does *not* mean, however, that these patents are always construed broadly. As in with the Wang and Pinpoint patents discussed above, the Federal Circuit often interprets the claims narrowly. Often—but not always (e.g., Freeny)!—the judges seem to take a pragmatic approach and narrow these broad patents. However, although this practice might save defendants in many of these trials, it only further undermines the notice function: now, not only are there claims with uncertain boundaries, the court adds additional unpredictability over how these claims will be interpreted if brought to trial and appeal. In effect, the enablement and *Alappat* decisions let the abstraction genie out of the bottle; the Federal Circuit appears to be attempting to stuff it back in the bottle using "creative claim construction" on a case-by-case basis. This situation cannot last.

9.3.3 Obvious Software Patents

Problems of abstraction are severely aggravated by the proliferation of large numbers of patents on trivial inventions. Arguably, the Wang and Pinpoint patents above (and possibly the Freeny patent) are at best trivial improvements on existing knowledge; at worst, they are blatantly obvious. The potential combination of large numbers of trivial patents with abstract claims creates enormous notice problems for software patents.

As we discussed in Chapter 3, patent proliferation is a general problem, one not restricted to software patents. Changes in the non-obviousness requirement for patentability have affected all types of patents. However, it is possible that features of software technology make it particularly susceptible to the patenting of obvious ideas, especially given the legal doctrines of non-obviousness developed by the Federal Circuit. For one thing, the general purpose nature of software technology—again, because the technology is abstract, similar techniques can be used in a wide range of applications—means that techniques known in one realm might be applied in another, yet the documentary evidence that the Federal Circuit requires for a demonstration of obviousness might not be published. For example, a large number of patents cover well-known business processes updated with a software implementation. It may not be a coincidence that several of the precedents where the Federal Circuit has weakened the non-obviousness requirement involve software (*In re Zurko, In re Lee*). Whatever the cause, the combination of large numbers of software patents that are both trivial and abstract produces significant problems of patent notice.

The effect of this flood is apparent in e-commerce patents. David M. Martin estimates that "if you're selling online, at the most recent count there are 4,319 patents you could be violating. If you

19 WMS Gaming Inc. v. Int'l Game Tech., 184 F.3d 1339 (Fed. Cir. 1999), Harris Corp. v. Ericsson, Inc. 417 F.3d 1241, 1253 (Fed. Cir. 2005)

20 Some recent exceptions are Lizardtech, cited above, DE Technologies v. Dell, 428 F. Supp. 2d 512, where the patent was held to be indefinite, and Ziarno v. Am. Nat'l Red Cross, 55 Fed. Appx. 553 which was held to have an inadequate written description.

also planned to advertise, receive payments for or plan shipments of your goods, you would need to be concerned with approximately 11,000."[21] One software executive estimates that checking clearance costs about $5,000 per patent.[22] It is no accident that most software users do not clear rights. Checking thousands of patents is clearly infeasible for almost any software product. Consequently, firms do not clear their technologies, they inadvertently infringe and costly litigation is the result.

9.4 Conclusion

Patents on software are *not* just like other patents. The evidence shows that software patents are particularly prone to litigation and to disputes over patent boundaries, a concern that has been raised about them since the 1960s. We attribute these problems to the abstract nature of software technology; too many software patents claim all technologies with similar form or all means of achieving a result, when the actual invention is much more limited and often trivial.

Patent law has developed a number of doctrines to circumscribe abstract patent claims. Unfortunately, the Federal Circuit has set software-specific precedents that essentially remove most restrictions on abstract claims in software. Perhaps the court acted out of a desire to promote patents in this field of technology that has historically not used patents. The result has been both a proliferation of software patents and a proliferation of lawsuits.

Software patents are not the only patents to suffer problems of abstract claims. Any technology can be claimed abstractly and, to make matters worse, the Federal Circuit has recently eroded limits on abstract patents for non-software business processes and even basic scientific ideas (e.g., *Laboratory Corp. of America v. Metabolite Laboratories*). But overall, software patents likely have a far greater influence on the performance of the patent system.

Software patents are, in fact, responsible for a major share of patent lawsuits. They thus play a central role in the failure of the patent system as a whole. Any serious effort at patent reform must address these problems and failure to deal with the problems of software patents—either with software-specific measures or general reforms—will likely doom any reform effort. We turn to possible changes in patent policy in the next chapter.

Advanced Topic: Abstraction and related doctrines

Morse's telegraph illustrates the problem of abstract patents and the use of patent law doctrines to restrict abstract claims. This invention falls into a variety of classes of technology that we list here in order of decreasing abstraction:

1. Telegraphy. Telegraphy is the transmission of information over distances without physical transport. Telegraphy is an ancient idea and ancient telegraphic inventions included communication by lights, smoke signals and semaphores.

2. Electrical telegraphy, the communication of information by electrical signals. The idea of electrical telegraphy dates back at least to 1753. An invention using electrostatic means (sparks) was demonstrated in 1795. Inventions using electrochemical means (generation of bubbles or discoloring impregnated paper) were demonstrated as early as 1809.

21 David Streitfeld, Los Angeles Times, February 8, 2003.
22 Mark Webbink, Red Hat Software, "Software Patents and Reality," presented November 17, 2006 at BU School of Law, http://www.researchoninnovation.org/swconf/webbinkslides.pdf.

3. Electromagnetic telegraphy. In 1820 Oersted demonstrated that electric current through a wire caused a magnetized needle to deflect. Based on the testimony of the eminent scientist Joseph Henry, the court in *O'Reilly v. Morse* (56 U.S. 62) concluded, "it was believed by men of science that this newly-discovered power might be used to communicate intelligence to distant places. And before the year 1823, Ampere of Paris, one of the most successful cultivators of physical science, proposed to the French Academy a plan for that purpose."

   There were, however, a number of technical obstacles to developing a practical invention based on this principle. Scientists soon found that the signal became weaker with distance and would not work over 200 feet. In 1831, Joseph Henry devised a method for using higher voltage and winding multiple coils on an electromagnet. This permitted him to demonstrate a system to his class at Princeton over greater distances. In 1835 Henry also devised the relay, so that multiple circuits could be combined to extend the range of the total system indefinitely. Pavel Schilling developed the first binary system of transmission (a code designated by whether the current was flowing or not) in Russia in 1832. Another electromagnetic telegraph was developed by the mathematician Gauss with Weber in Germany in 1833. And the first commercial telegraph system was introduced by Wheatstone and Cooke in England in 1837. This system used five wires, allowing 32 different characters to be designated by the state of each circuit.

4. Morse's specific invention (described in patents granted in 1840, 1846 and re-issued in 1848). Morse's invention combined features developed by others and added some of his own. Morse used electromagnets with coils and relays, both developed by Henry. Morse used a single wire, rather than five wires. He did this by developing a code that designated alphanumeric characters by combinations of binary signals of different duration. Also, Morse developed an elaborate printing device (which was later largely abandoned in favor of sound).

In this hierarchy, the more abstract classifications are broader, that is, they include a larger range of inventions; the more specific classes are subsets of the more abstract classes. In addition—and this is an important point—abstract classes may include many potential inventions that were not known at the time. For example, the term "electromagnetic telegraphy" might reasonably be interpreted to include the use of electricity and magnetism to transmit images, not just text. This was likely seen as a desirable application of the technology at the time of Morse's application in 1838 and earlier, but it was not until 1843 that Alexander Bain invented the first primitive facsimile machine. So an abstract principle or an abstract inventive idea describes some general characteristics of the function or structure of a class of inventions that may include some potential applications for which specific means of realizing the desired results have not yet been invented.

What then is wrong with granting patents on such abstract principles? What is wrong, say, with having a patent that claims to cover telegraphy by electromagnetism? In theory such patents might seem an attractive way to provide incentives to scientists like Oersted.[23] However, this ignores the requirements of an efficient property system: property requires clear notice. Lacking a tangible manifestation of the "invention," courts might find it difficult to determine priority or infringement. For example, who first conceived the idea of electromagnetic telegraphy? It was certainly not Morse. There is documentation that Ampere had the idea in 1823, but he might not have been the first. The idea of electromagnetic telegraphy may well have occurred to many people, including Oersted himself, more or less simultaneously upon hearing of Oersted's accidental discovery. Clearly it is much easier for a

23 Oren Bar-Gill and Gideon Parchomovsky (2006) present a different argument. in "A Marketplace for Ideas?," 84 Texas Law Review 397.

court to determine priority when the act of invention requires a tangible demonstration or a working scale model (as was the case during much of the 19th century).

But even if priority is determined by a concrete physical invention, it is still possible to claim abstract ideas that may have obscure boundaries. Indeed, when Morse obtained a re-issue of his patent in 1848 he added just such an abstract claim to the seven specific claims that described his actual telegraph:

> "Eighth. I do not propose to limit myself to the specific machinery, or parts of machinery, described in the foregoing specifications and claims; the essence of my invention being the use of the motive power of the electric or galvanic current, which I call electro-magnetism, however developed, for making or printing intelligible characters, letters, or signs, at any distances, being a new application of that power, of which I claim to be the first inventor or discoverer."

There are two problems with such an abstract claim. First, such a claim rewards Morse for inventions he did not invent. As the Supreme Court wrote,

> For aught that we now know some future inventor, in the onward march of science, may discover a mode of writing or printing at a distance by means of the electric or galvanic current, without using any part of the process or combination set forth in the plaintiff's specification. His invention may be less complicated -- less liable to get out of order -- less expensive in construction, and in its operation. But yet if it is covered by this patent the inventor could not use it, nor the public have the benefit of it without the permission of this patentee.

Such future inventors would face *reduced* incentives because they would have to obtain a license from Morse to develop their inventions.

Second, it may be difficult to determine the boundaries of such a claim and thus it may be difficult to provide notice, to conduct clearance searches, or to even determine what the prior art is. For example, Morse's first seven claims describe his actual invention and the words in those claims are given meaning by reference to the description of that invention. Given this description, one can readily understand how to make and use Morse's telegraph and this specific knowledge can be compared to the ways other technologies are made and used. But no such information is provided for the eighth claim. No information is provided on how to make and use all the inventions covered by this claim because, in fact, Morse himself did not know all of these inventions or how they work. Consequently, this claim lacks clear boundaries. What technologies are included in "making intelligible signs at any distance"? Does this include signs made with pointing needles (discovered before Morse)? Does it include handwritten signs used in Bain's facsimile machine? Does it include video images (cathode ray tubes work by exerting electromagnetic force on electrons)? Does it include signs made with electromagnetic radiation, e.g., beacons or semaphores using light (an interpretation Morse could not have been expected to know)? Does it matter how electromagnetism is used? For example, Bain patented a facsimile machine that used electromagnets, but only to synchronize a timing mechanism (which he had patented earlier) at two locations. Does this count as electromagnetism "however developed"? Similarly, does Morse's eighth claim cover wireless electromagnetic communication as in radio telegraphy? Indeed, is there any limitation on how the information is communicated to the distant location? Although some technologies clearly fall within the scope of this claim, in other cases the boundaries are not so clear. Abstract words that provide only a partial functional description of a technology class are simply insufficient to provide clear notice in many cases.

The Morse decision is often described as based on lack of an adequate disclosure. However, this is a specific case where the disclosure was inadequate because the claim was abstract and therefore involved unknown and uninvented technologies. It may be helpful to distinguish this doctrines from several closely related doctrines in patent law. Sometimes *O'Reilly v. Morse* is discussed as an example of the enablement doctrine of "undue experimentation" discussed in the case of Sawyer and Man in Chapter 3 (e.g., Merges and Nelson 1990, p. 850). However, there is an important difference between these cases. With Sawyer and Man, the class of claimed inventions was identifiable and the court could therefore consider the amount of experimentation needed to find an effective fiber for electric lighting. However, in Morse, the class of claimed inventions was not known. The court could not determine the amount of experimentation needed to find make and use the claimed inventions, because these inventions were not identified. Instead, the court had to make a judgment about the abstractness of the claim—their reasoning was that the claim was so abstract that there would *likely* be inventions covered by the claims that were not described in the patent. This is clearly a more difficult and less fact-based inquiry.

A related issue is the balance between the incentives to pioneer inventors and the incentives to commercialize their inventions. Some scholars advocate granting broad patents to pioneers to encourage them to commercialize their inventions (see the "prospect" theory of Kitch 1977, and also Kieff 2001). More generally, the Morse decisions is sometimes described as primarily about setting the breadth of the patent in order to adjust balance between pioneer inventors and subsequent commercializers or other inventors. Although the decision in Morse did affect the scope of the patent, it had little to do with this balance. Because the inventions covered by Morse's eighth claim were not known, the grant of a broad patent covering them to Morse could hardly provide any incentive for him or for anyone else to commercialize these inventions. The problem with Morse's eighth claim was not so much that it was broad, but that the range of inventions it covered was not known.

Two other distinctions are also worth noting. Although some abstract patents use vague words such as "material object," others do not, such as the words in the Wang patent. Sometimes the indefiniteness requirement discussed in Chapter 3 is interpreted to relate to the vagueness of words used in claims. The point about abstract patents is not so much that they use vague words (although they lend themselves to that abuse), but that the words circumscribe a set of technologies that is not known. Finally, abstraction is sometimes considered opposite to the practical (as in *Alappat*). However, Morse's patent clearly had practical value, yet the eighth claim was also abstract.