claim element is *not* interpreted to cover every means of performing the function. Instead, the courts apply a different rule of claim construction, limiting the scope of these claims by reading in the particular technologies described in the patent specification.[53] To take an example, suppose that the patent claim includes as an element a "means for processing data."[54] Read literally, without reference to Section 112(f), this language would encompass any possible means for processing data, including any computer, but also a calculator, an abacus, pencil and paper, and perhaps even the human brain. Section 112(f) permits the use of such functional language but doesn't permit it to cover any means of performing the data-processing function. Instead, the claim would be limited to the particular "means for processing data" actually described in the patent specification (say, an iPad) "and equivalents thereof."[55]

This "means-plus-function" claiming is not limited to patent claims covering machines or articles of manufacture. The statute speaks of "structure, material, *or acts* in support" of the function,[56] a clear indication that the concept applies to process claims as well. And indeed, the courts have applied the same basic rules to so-called "step-plus-function" claims.[57] Like machine claims defined in functional terms, step-plus-function claims prevent process patentees from claiming the function itself, limiting them to the particular algorithm or series of steps disclosed in the specification to perform that function "and equivalents thereof."[58]

---

53. *See In re Hyatt*, 708 F.2d 712, 713–14 (Fed. Cir. 1983).

54. Another limit on means-plus-function claiming is that it must occur in the course of a combination of elements. "Single means" claims are invalid. *See id.* at 714. If there is more than one element, however, each of the elements can itself be a means-plus-function claim.

55. *See, e.g.*, *In re Donaldson Co.*, 16 F.3d 1189, 1193–94 (Fed. Cir. 1994) (en banc).

56. 35 U.S.C. § 112(f) (2012) (emphasis added).

57. In this paper, I will sometimes use the term "means-plus-function" to encompass both true means-plus-function claims to machines and step-plus-function claims to processes.

58. *Alloc, Inc. v. U.S. Int'l Trade Comm'n*, 342 F.3d 1361, 1373 (Fed. Cir. 2003) (quoting 35 U.S.C. § 112 ¶ 6 (2000)); *O.I. Corp. v. Tekmar Co.*, 115 F.3d 1576, 1583 (Fed. Cir. 1997). At the same time, courts have cautioned that not every step that includes "an 'ing' verb" should be construed as a step-plus-function claim. *Id.* at 1582–83. And one judge has gone further, asserting in a concurrence that step-plus-function claims "require distinct analysis" from means-plus-function claims. *Seal-Flex, Inc. v. Athletic Track & Court Constr.*, 172 F.3d 836, 848 (Fed. Cir. 1999) (Rader, J., concurring). But the differences Judge Rader identifies relate to identifying step-plus-function claims, not to how they are treated once they are identified. There is a surprising dearth of case law on step-plus-function claims. *See* Brad A. Schepers, Note, *Interpretation of Patent Process Claims in Light of the Narrowing Effect of 35 U.S.C.*

While the last phrase in the statute—"and equivalents thereof"—permits some broadening of both means-plus-function and step-plus-function claims,[59] courts in the last fifteen years have not read "equivalents" broadly.[60]

The result is that means-plus-function claiming today is viewed as narrow and easy for potential infringers to evade. Patent lawyers tend to avoid means-plus-function claim language, except as an "extra" put in a separate claim to hedge risk.[61] Litigators tend to dismiss those claims, reasoning that once the defendant is allowed to read limits in from the specification, there will always be a way to avoid infringement.[62] In short, while the 1952 Act theoretically restored functional claiming, the option it offered was not really functional claiming at all and has not

---

§ 112(6), 31 IND. L. REV. 1133, 1163 (1998).

59. *See, e.g.*, *WMG Gaming, Inc. v. Int'l Game Tech.*, 184 F.3d 1339, 1347 (Fed. Cir. 1999); *Al-Site Corp. v. VSI Int'l, Inc.*, 174 F.3d 1308, 1320–21 (Fed. Cir. 1999); *Chiuminatta Concrete Concepts, Inc. v. Cardinal Indus., Inc.*, 145 F.3d 1303, 1310 (Fed. Cir. 1998).

60. *See, e.g.*, John R. Allison & Mark A. Lemley, *The (Unnoticed) Demise of the Doctrine of Equivalents*, 59 STAN. L. REV. 955 (2007). For more discussion, *see infra* notes 196–201 and accompanying text.

61. Note, *Everlasting Software*, 125 HARV. L. REV. 1454, 1460 n.38 (2012) ("[P]atent attorneys often avoid means-plus-function claiming . . . ."). Dennis Crouch finds that the number of claims with "means for" language has declined from 24% in 2001 to only 7% today. Dennis Crouch, *Means Plus Function Claiming*, PATENTLY-O, (Jan. 14, 2013), http://www.patentlyo.com/patent/2013/01/means-plus-function-claiming.html. And that overstates their use, since most of these claims are in patents that also include other claims without that language. For a discussion of the specifics of means-plus-function claiming in software, see Sebastian Zimmeck, *Use of Functional Claim Elements for Patenting Computer Programs*, 12 J. HIGH TECH. L. 168 (2011).

62. *See, e.g.*, Ronald L. Lacy et al., *Crafting the Claims*, *in* ELECTRONIC AND SOFTWARE PATENTS: LAW AND PRACTICE (2d ed. 2011) ("Like method claims, apparatus claims may be afforded a broader scope of interpretation than means-plus-function claims. The apparatus claim is interpreted in light of the specification, but not under" Section 112[f].); Rudolph P. Hofmann, Jr. & Edward P. Heller, III, *The Rosetta Stone for the Doctrines of Means-Plus-Function Patent Claims*, 23 RUTGERS COMP. & TECH. L.J. 227, 231 (1997) ("Thus, while general claims enjoy a scope as broad as their unambiguous claim language permits, means-plus-function claims are given a different, more limited treatment."); Michael A. Molano & Graham (Gray) M. Buccigross, *Traps for the Unwary: Issues Surrounding Means-Plus-Function Claims in the Software Context*, *in* FUNDAMENTALS OF PATENT PROSECUTION 2011: A BOOT CAMP FOR CLAIM DRAFTING AND AMENDMENT WRITING (Practising Law Institute 2011); Ryan Sharp, *Can Beauregard Claims Show You the Money?*, 2 CYBARIS: INTELL. PROP. L. REV. 25, 34 (2011), ... http://web.wmitchell.edu/cybaris/wp-content/uploads/2011/08/Sharp.pdf ("[I]t is well known that means-plus-function claims are narrowly construed . . . ."). For a dissenting view, see Zimmeck, *supra* note 61, at 228–29 (arguing that means-plus-function claims can be broader than corresponding apparatus claims that disclose structure).

been viewed as an attractive option for those seeking broad patent claims.

II. THE NEW FUNCTIONAL CLAIMING

While means-plus-function claiming under Section 112(f) is in disfavor among patentees, that doesn't mean inventors have stopped seeking broad patent claims. One way to seek broad patent claims is to try to define a broad group of things. That works reasonably well in chemistry or biotechnology, where we have a standard language that allows us to define groups and determine whether a later-developed chemical is in the group. But in other areas, like mechanical inventions, a broad claim requires defining the invention at a higher level of abstraction—as hook-and-eye closures generally rather than the particular implementation of Velcro, or as hybrid gasoline-electric engines generally rather than the particular implementation of that concept in, say, Honda's Integrated Motor Assist. While these abstract claims are broader—they encompass a genus of possible implementations—they still require an irreducible minimum structure, and that structure limits the claim. There may be a number of different hook-and-eye closures, but means for attaching that don't include hooks and eyes don't fall within the scope of a patent claim that requires hook-and-eye closures.

Computer software gives patentees the opportunity to take abstraction in patent claiming to an extreme.[63] For the genius of computers is that structure and function can be almost completely separated. The hardware "structure" of a computer software invention is . . . a computer. Generally speaking it doesn't much matter what type of computer a program runs on; all computers have standard elements. That fact has given patentees an opening to write "structural" claims in which the structure is not novel and does no work. A patentee who claims "means for calculating an alarm limit" is invoking the limits of Section 112(f), and the claim will accordingly be limited to the particular software algorithm or implementation the patentee described in the specification. But if the same patentee claims "a computer programmed to calculate an alarm limit," courts today will read the term "computer" as sufficient structure and will understand the claims to cover any computer that can calculate an alarm limit, however the calculation is programmed.[64]

---

63. *See* Note, *supra* note 61, at 1459–60.

64. One might argue that such a claim is really a process claim, not a system claim at all. The Federal Circuit drew that conclusion in *Cybersource Corp. v. Retail*

Modern software patent claims quite commonly take this form. Indeed, by my estimate there are tens, perhaps hundreds of thousands of such patents.[65] Here are just a few examples from litigated software cases:

- "A method for generating a file note for an insurance claim, comprising the steps . . . executed *in a data processing system*, of [a series of conceptual steps]."[66]

- "A method for operating *a computer system* to facilitate an exchange of identities between two anonymous parties, comprising the steps of [a number of process steps]."[67]

- "A *computer readable medium* containing program instructions for detecting fraud in a credit card transaction between a consumer and a merchant over *the Internet,* wherein execution of the program instructions by *one or more processors of a computer system* causes the one or more processors to carry out the steps of [a number of steps]."[68]

- "*A computer program product* for use in *a system having at least one client workstation and one network server coupled to said network environment*, wherein said network environment is a distributed hypermedia environment, the computer program product comprising: *a computer usable medium* having computer readable program code physically

---

Decisions, Inc., 654 F.3d 1366, 1376–77 (Fed. Cir. 2011). But since Section 112(f) applies to process as well as system claims, the characterization should not matter for our purposes.

65. Colleen Chien and Aashish Karkhanis estimate based on a study of litigated patents that 100% of troll software patents and 50% of non-troll software patents use functional claiming. Of those patents, 40% had only functional abstractions in the specification, while the rest had at least some code or structural definition. Colleen V. Chien & Aashish R. Karkhanis, *Software Patents and Functional Claiming* 40 (Santa Clara Univ. Sch. of Law, Legal Studies Research Papers Series, Working Paper No. 06-13, 2013), *available at* http://papers.ssrn.com/sol3/papers.cfm?abstract_id=2215867.

66. U.S. Patent No. 7,017,111 B1 col.7 ll.27–29 (filed Apr. 14, 2000) (emphasis added), at issue in *Accenture v. Guidewire, Inc.*, No. 11-1486 (Fed. Cir. Sept. 5, 2013). Full disclosure: I represent Guidewire in this case.

67. U.S. Patent No. 5,884,270 col.23 ll.60–62 (filed Sept. 6, 1996) (emphasis added), at issue in *Walker Digital, LLC v. MySpace, Inc.*, No. 11-318 (D. Del. July 25, 2013). Full disclosure: I represented LinkedIn in this case.

68. U.S. Patent No. 6,029,154, col.4 ll.45–51 (filed July 28, 1997) (emphasis added), at issue in *Cybersource*, 654 F.3d 1366.

embodied therein, said computer program product further comprising [software steps]."[69]

- "*A computer-readable storage medium* storing program code for causing *a server* that serves as a gateway to a client to perform the steps of: [processing program instructions]."[70]

- "A *computer aided* method of managing a credit application, the method comprising the steps of: receiving credit application data from *a remote application entry and display device*; selectively forwarding the credit application data to *remote funding source terminal devices*; [other steps without hardware omitted]."[71]

- "A method for distribution of products over *the Internet* via a facilitator, said method comprising the steps of [many steps that do not require any hardware]."[72]

- "A method using *a computer network and a database* accessible through the *computer network*, comprising the steps of [various steps, some employing the terms "computer network" and "database"]."[73]

- "[C]omputer readable program code configured to cause a *computer* to [perform a series of display steps]."[74]

- "In *a data communication system* wherein messages comprising data code words are to be transmitted from *a data transmitter* to one or more of *a plurality of data*

---

69. U.S. Patent No. 5,838,906 col.17 ll.58–65 (filed Oct. 17, 1994) (emphasis added), at issue in *Eolas Techs., Inc. v. Microsoft Corp.*, 457 F.3d 1279 (Fed. Cir. 2006).

70. U.S. Patent No. 6,092,194 col.13 ll.13–15 (filed Nov. 6, 1997) (emphasis added), at issue in *Finjan, Inc. v. Secure Computing Corp.*, 626 F.3d 1197 (Fed. Cir. 2010).

71. U.S. Patent No. 7,181,427 B1 col.20 ll.54–59 (filed Sept. 3, 1997) (emphasis added), at issue in *Dealertrack, Inc. v. Huber*, 674 F.3d 1315 (Fed. Cir. 2012).

72. U.S. Patent No. 7,346,545 B2 col.8 ll.5–6 (filed May 29, 2001) (emphasis added), at issue in *Ultramercial, LLC v. Hulu, LLC*, 657 F.3d 1323 (Fed. Cir. 2011).

73. U.S. Patent No. 5,778,367 col.12 ll.37–59 (filed Dec. 14, 1995) (emphasis added), at issue in *MySpace, Inc. v. Graphon Corp.*, 672 F.3d 1250 (Fed. Cir. 2012).

74. U.S. Patent No. 5,878,400 col.23 ll.22–23 (filed June 17, 1996) (emphasis added), at issue in *Versata Software Inc. v. SAP Am., Inc.*, 2011 WL 4017939 (E.D. Tex. Sept. 9, 2011).

receivers, a method of transmission of such messages comprising the steps of [data processing steps]."[75]

- "A method for use in *a computer having a display* comprising the steps of [employing various software tools]."[76]

- "[S]oftware executing in the *central processor* to configure the processor so as to [perform certain functions]."[77]

Nor is this a problem only for old patents currently in litigation. Patents issuing today have the same sorts of problems.[78]

---

75. U.S. Patent No. 4,975,952 col.7 ll.47–51 (filed May 11, 1988) (emphasis added), at issue in *Fujitsu Ltd. v. Netgear Inc.*, 620 F.3d 1321 (Fed. Cir. 2010).

76. U.S. Patent No. 4,763,356 col.17 ll.25–26 (filed Dec. 11, 1986) (emphasis added), at issue in *Lucent Techs., Inc. v. Gateway, Inc.*, 580 F.3d 1301 (Fed. Cir. 2009).

77. U.S. Patent No. 7,698,372 B2 col.43 ll.53–54 (filed Sept. 28, 2009) (emphasis added), at issue in *Easyweb Innovations, LLC v. Twitter Inc.*, No. 11-CV-04550 (E.D.N.Y. filed Sept. 19, 2011). Full disclosure: I represent Twitter in this case, which is pending.

78. Here are three examples selected from ten patents I reviewed that issued in January 2013:

- "[A] computer programmed to receive input data from the job-site positioning system and the feature locating system, the computer evaluating the input data and a relationship between the job-site positioning system and the feature locating system to determine the location of the topographic feature at the job-site." U.S. Patent No. 8,363,210 B2 col.6 ll.44–49 (filed Oct. 26, 2007) (issued Jan. 29, 2013).

- "A method of creating a light effect, said effect providing perceived moving images to the human eye, said method comprising:
  i. concurrently exposing a variety of colored designs to at least two different colors of light emitting diode lights;
  ii. controlling said light emitting diode lights to provide a desired light effect selected from the group consisting of
    i. movement, and,
    ii. complete change of image colors in the designs."

  U.S. Patent No. 8,350,481 B1 col.2 ll.43–51 (filed May 27, 2010) (issued Jan. 8, 2013).

- "[S]aid programmable integrated circuit being programmable to accept said data as input and generate an output signal to said LED driver circuit corresponding to said data." U.S. Patent No. 8,344,639 B1 col.7 ll.47–50 (filed Jan. 31, 2010) (issued Jan 1, 2013).

My point in highlighting these examples is not to suggest that all these claims are unduly broad, though some have been held invalid and some others probably will be or should have been. Some of these claims contain process steps sufficiently detailed that the resulting claims are quite narrow. Rather, the point is that the claims are effectively unlimited *as a matter of structure*. The function they perform may be simple or complex, broad or narrow, but in the modern world the patent claims listed above effectively cover *any device* that performs that function in any way. Even if it were theoretically possible to implement a computer program in some device other than "a computer having a display," as a practical matter, any use of the steps specified in that patent is going to occur in a computer, and any modern computer is going to have a display. As a practical matter, claims with a trivial structural element that everyone must include are claims to function, not structure.

The absence of a real hardware limitation wouldn't be a problem if the patentee's claims were limited to a particular software implementation of the invention. Arguably, we shouldn't care what hardware substrate a software invention runs on. In fact, however, those claims are rarely limited to a particular software algorithm. The process steps implemented in the generally claimed computer are also claimed in broad functional terms. That is, the patentee claims the end it accomplishes, not the means of getting there. The presence of a nominal hardware limitation serves to obscure the fact that the real structure doing the work—the computer program—is absent.

Indeed, software patent claims often go further. Rather than claiming any implementation of a particular idea in a computer, these "capability claims" assert ownership of any device that is capable of implementing that idea, whether or not the device actually does so. There are numerous examples of claims reciting phrases such as "programmable selection means for . . . ,"[79] ". . . capable of engaging,"[80] "adapted to . . . ," "for . . . -ing," "operable to . . . ," and the like. While any of a variety of language constructs may be recited by patentees to denote capability literally present, a recent sample of patent claims issued indicates that even the most overt form ("capable of") appears *in the claims* of nearly twelve thousand patents issued in the first nine months of 2011.[81] When compared to patents issued a decade earlier,

79. U.S. Patent No. 4,685,084 col.6 l.8 (filed June 7, 1985), at issue in *Intel Corp. v. U.S. Int'l Trade Comm'n*, 946 F.2d 821, 824 n.3 (Fed. Cir. 1991).

80. U.S. Patent No. US RE37,545 E col.6 ll.56–57 (filed Oct. 21, 1998), at issue in *Revolution Eyewear Inc. v. Aspex Eyewear Inc*., 563 F.3d 1358, 1362 (Fed. Cir. 2009).

81. Mark A. Lemley, David W. O'Brien, & Wade Malone, Capability Claiming n.13 (2011) (unpublished manuscript), *available at* https://www.law.stanford.edu/sites/

numbers and percentages are essentially unchanged.[82] Overwhelmingly, these capability claims are software or computer technology patents. And while the Federal Circuit has read these claims to require the technology to be programmed into the system, as opposed to covering computers that would have to be reprogrammed to perform the identified function,[83] the combination of a structural element that is essentially not limiting and a function that doesn't even have to be enabled can lead to patents that are broad indeed.[84]

Software patents, then, have brought back functional claiming as it existed before 1952. The computer hardware elements impose no real limitation on an invention that must, of necessity, be implemented in a computer, particularly since one of the features of computer technology is that the particular hardware chosen usually doesn't constrain what software can be run. Thus, as a practical matter the only real limits on claims of this sort are the steps the software must perform.

Those software steps are quite often defined in functional terms. The software claim elements generally do not specify particular coding approaches or modules that must be used, much less the code that implements those modules.[85] Indeed, courts have not required significant

---

default/files/event/266396/media/slspublic/Panel%201%20-%20Mark%20Lemley,%20et%20al%20-%20Capability%20Claiming.pdf.

> Specifically, between 1-January-2011 and 14-October-2011, a search of the USPTO Patent Full-Text and Image Database (http://patft.uspto.gov/) indicates that a total of 11,746 U.S. Patents (including reissues) granted with the textual string "capable of" at least once in the claims. With 193,507 U.S. Patents issued during the same period, that is slightly more that 6% of the total. Inclusion of "adapted to" in the search, more than doubles the number of hits to 27,393 (or more the 14% of all patents issued calendar year 2011 to date).

Id. at n.13.

82. *Id.* at n.14 ("12,343 (or 6.5%) of 184,045 U.S. Patents issued in calendar year 2001 include the textual string 'capable of' at least once in the claims.").

83. *Typhoon Touch Techs., Inc. v. Dell, Inc.*, 659 F.3d 1376, 1382 (Fed. Cir. 2011); *Fantasy Sports Props. v. Sportsline.com, Inc.*, 287 F.3d 1108, 1118 (Fed. Cir. 2002).

84. If read that broadly, they are probably also invalid, since any processor is presumably "capable of" being programmed to perform the steps in question.

85. Robin Feldman argues that functional claiming in software results in part from early judicial doubts about the patenting of computer algorithms themselves:

> The message was clear, however, that if an innovation was ever going to survive a court challenge, it had to avoid being labeled an algorithm or looking too much like math. The result was an attempt to describe the process of what was happening in simple English terms by moving the description of the process to an even more abstract plane. If successful, the approach would have the advantage of allowing the inventor to tie up an even larger swath of territory, given that broad, abstract language had the potential to cover many

disclosure of code or program structure even in the specification.[86] Instead, the software elements tend to be drafted in terms of the function

---

> different ways of accomplishing the same result. For example, an algorithm designed to operate on digital images may be claimed by the simple language of what it is intended to do, thus covering a far wider territory than mathematically describing the algorithm itself.

ROBIN FELDMAN, RETHINKING PATENT LAW 109 (2012). Feldman continues:

> Most troubling, the incentive to describe what is happening in linguistic rather than mathematical terms could also provide a tremendously wide footprint for each patent. For example, consider the applicant who would now simply use the claims language "applying a statistical model" rather than providing the notation of the actual statistical model or formula that is used. The general term "statistical model" will have very broad coverage if it is not strictly defined.

Id. at 111–12.

86. For instance, the Federal Circuit has held that software patentees need not disclose source or object code, flow charts, or detailed descriptions of the patented program. Rather, the court has found high-level functional descriptions sufficient to satisfy both the enablement and best mode doctrines. *See Fonar Corp. v. General Elec. Co.*, 107 F.3d 1543, 1549 (Fed. Cir. 1997); *In re Hayes Microcomputer Prods., Inc. Patent Litig.*, 982 F.2d 1527, 1533–34 (Fed. Cir. 1992); *Northern Telecom, Inc. v. Datapoint Corp.*, 908 F.2d 931, 943 (Fed. Cir. 1990). *See also* Lawrence D. Graham & Richard O. Zerbe, Jr., *Economically Efficient Treatment of Computer Software: Reverse Engineering, Protection, and Disclosure*, 22 RUTGERS COMPUTER & TECH. L.J. 61, 96–97 (1996); Greg R. Vetter, *Patent Law's Unpredictability Doctrine and the Software Arts*, 76 MO. L. REV. 763 (2011) (criticizing this low standard); Anthony J. Mahajan, Note, *Intellectual Property, Contracts, and Reverse Engineering After* ProCD*: A Proposed Compromise for Computer Software*, 67 FORDHAM L. REV. 3297, 3317 (1998). For example, in *Northern Telecom*, the court noted expert testimony that various programs could be used to implement the invention, and that it would be "relatively straightforward [in light of the specification] for a skilled computer programmer to design a program to carry out the claimed invention." *Northern Telecom*, 908 F.2d at 941–42. The court continued:

> The computer language is not a conjuration of some black art, it is simply a highly structured language . . . . [T]he conversion of a complete thought (as expressed in English and mathematics, i.e. the known input, the desired output, the mathematical expressions needed and the methods of using those expressions) into a language a machine understands is necessarily a mere clerical function to a skilled programmer.

Id. at 942 (quoting *ex rel. Sherwood*, 613 F.2d 809, 817 n.6 (C.C.P.A. 1980). And in *Fonar Corporation v. General Electric Company*, the court explained:

> As a general rule, where software constitutes part of a best mode of carrying out an invention, description of such a best mode is satisfied by a disclosure of the functions of the software. This is because, normally, writing code for such software is within the skill of the art, not requiring undue experimentation, once its functions have been disclosed. It is well established that what is within the skill of the art need not be disclosed to satisfy the best mode requirement as long as that mode is described. Stating the functions of the best mode software satisfies that description test. We have so held

they perform, claiming things like "program code for causing a server that serves as a gateway to a client to perform the steps of" a, b, and c.[87] Any code that causes the computer to perform those steps infringes the patent claim. It is the function, not the particular tool the patentee developed to perform the function, that is the subject of the patent.

Nonetheless, these functional software claims have not been subject to the normal constraints Section 112(f) imposes on means-plus-function claims. While the Federal Circuit has of late been quite vigilant in limiting software patentees who write claims in means-plus-function format to the particular algorithms that implement those claims,[88] it has

---

> previously and we so hold today. Thus, flow charts or source code listings are not a requirement for adequately disclosing the functions of software.

Fonar, 107 F.3d at 1549 (citations omitted).

Indeed, in a few cases the Federal Circuit has gone so far as to hold that patentees can satisfy the written description and best mode requirements for inventions implemented in software even though they do not use the terms "computer" or "software" anywhere in the specification! *See Robotic Vision Sys., Inc. v. View Eng'g, Inc.*, 112 F.3d 1163 (Fed. Cir. 1997) (best mode); *In re Dossel*, 115 F.3d 942 (Fed. Cir. 1997) (written description).

By contrast, in *White Consol. Indus., Inc. v. Vega Servo-Control, Inc.*, 713 F.2d 788 (Fed. Cir. 1983), the Federal Circuit had invalidated a patent for a machine tool control system which was run by a computer program. Part of the invention was a programming language translator designed to convert an input program into machine language, which the system could then execute. The patent specification identified an example of a translator program, the so-called SPLIT program, which was a trade secret of the plaintiff. *Id.* at 789. The court held that the program translator was an integral part of the invention, and that mere identification of it was not sufficient to discharge the applicant's duty under Section 112. *Id.* at 790. The court seemed concerned that maintaining the translator program as a trade secret would allow White to extend the patent beyond the seventeen year term then specified in the patent code. *Id.* at 791.

While *White* suggests that it is not sufficient merely to identify the program or its functions, more recent Federal Circuit authority is overwhelmingly to the contrary. *See, e.g.*, *In re Dossel*, 115 F.3d at 946–47 ("While the written description does not disclose exactly what mathematical algorithm will be used to compute the end result, it does state that 'known algorithms' can be used to solve standard equations which are known in the art." This was deemed sufficient to describe the invention). For discussion of this issue in more detail, see Dan L. Burk & Mark A. Lemley, *Is Patent Law Technology-Specific?*, 17 BERKELEY TECH. L.J. 1155 (2002).

87. U.S. Patent No. 6,092,104 col.13 ll.13–15 (filed Nov. 6, 1997), at issue in *Finjan, Inc. v. Secure Computing Corp.*, 626 F.3d 1197 (Fed. Cir. 2010).

88. *Function Media, LLC v. Google Inc.*, 708 F.3d 1310, 1318 (Fed. Cir. 2013); *ePlus, Inc. v. Lawson Software, Inc.*, 700 F.3d 509, 518–19 (Fed. Cir. 2012); *Noah Sys., Inc. v. Intuit Inc.*, 675 F.3d 1302, 1312–13 (Fed. Cir. 2012); *Ergo Licensing, LLC v. CareFusion 303, Inc.*, 673 F.3d 1361, 1362, 1365 (Fed. Cir. 2012); *Typhoon Touch Techs., Inc. v. Dell, Inc.*, 659 F.3d 1376, 1384–86 (Fed. Cir. 2011) (means-plus-function software claims required disclosure of corresponding structure performing that function in the specification, but that structure did not need to be described in the form of software code); *In re Aoyama*, 656 F.3d 1293, 1294, 1297–98 (Fed. Cir. 2011) (means-plus-function software patent claim invalid as indefinite for failure to disclose the

not treated any of the claims discussed above as means-plus-function claims at all. The presence of structure in the form of "a computer" or "a processor" or even "the Internet" has led the Federal Circuit to give these claims control over the claimed function however implemented.[89] As a

---

corresponding algorithm performing that function); *Aristocrat Techs. Austl. PTY Ltd. v. Int'l Game Tech.*, 521 F.3d 1328, 1337–38 (Fed. Cir. 2008); *WMS Gaming, Inc. v. Int'l Game Tech.*, 184 F.3d 1339, 1349 (Fed. Cir. 1999) ("[T]he disclosed structure is not the general purpose computer, but rather the special purpose computer programmed to perform the disclosed algorithm."). *Cf. HTC Corp. v. IPCom GmbH & Co., KG*, 667 F.3d 1270, 1272–73 (Fed. Cir. 2012) (where a means-plus-function software claim would have been invalid as indefinite for failure to disclose the algorithm that performed the functions of the software, but defendant waived the issue).

For discussion of these cases, see Sharon Barkume & Michael R. Bielski, *Strict Interpretation of 35 U.S.C. § 112: Requires Universities to Examine Their Patenting Methods*, 28 TOURO L. REV. 183, 197–98 (2012); Elise S. Edlin, *Computer Claim Disarray*: *Untangling the Means-Plus-Function Doctrine to Eliminate Impermissible Functional Claiming in Software Patents*, 28 BERKELEY TECH. L.J. 417, 419–20 (2013); Christa J. Laser, *A Definite Claim on Claim Indefiniteness: An Empirical Study of Definiteness Cases of the Past Decade with a Focus on the Federal Circuit and the Insolubly Ambiguous Standard*, 10 CHI.-KENT J. INTELL. PROP. 25, 37 tbls.6 & 7, 39–41 (2010); Vetter, *supra* note 86, at 797–99.

In addition, the Federal Circuit's current approach to written description under Section 112(a) also seems inconsistent with allowing functional claiming. There, the court has demanded that inventions be described in structural terms. *See Ariad Pharms., Inc. v. Eli Lilly & Co.*, 598 F.3d 1336, 1352 (Fed. Cir. 2010) (en banc); *Regents of the Univ. of Calif. v. Eli Lilly & Co.*, 119 F.3d 1559 (Fed. Cir. 1997) ("An adequate written description of a DNA . . . requires a precise definition, such as by structure, formula, chemical name, or physical properties") (internal quotation omitted). And while many have thought that rule applied only to biotechnology, *Ariad* denies any such limitation. *Id.* So it would seem that the court would be inclined to hold a software patent that only described function, not structure, invalid under the written description doctrine.

89. *See, e.g.*, *Inventio AG v. ThyssenKrupp Elevator Ams. Corp.*, 649 F.3d 1350, 1359–60 (Fed. Cir. 2011) (where "computing unit" connoted sufficiently definite structure that it did not invoke Section 112(f)); *LG Elecs., Inc. v. Bizcom Elecs., Inc.*, 453 F.3d 1364, 1372–73 (Fed. Cir. 2006), *rev'd on other grounds sub nom. Quanta Computer, Inc. v. LG Elecs., Inc.*, 553 U.S. 617 (2008) (holding a "claimed 'control unit' that comprised a 'CPU' and a 'portioned memory system' recited sufficiently definite structure to perform the recited 'controlling the communication unit' function"); *but see Brown v. Baylor Healthcare Sys.*, 381 F. App'x. 981, 983–84 (Fed. Cir. 2010) (finding that even if a "computing unit" is read to mean a computer, simply disclosing "a general processor without more" is not enough "to perform the claimed function" and avoid the application of Section 112(f)). The origins of this approach seem to be in the 1990s, when the Federal Circuit decided *In re Alappat*, 33 F.3d 1526 (Fed. Cir. 1994) (en banc). That court held:

> Alappat admits that claim 15 would read on a general purpose computer programmed to carry out the claimed invention, but argues that this alone also does not justify holding claim 15 unpatentable as directed to nonstatutory subject matter. We agree. We have held that such programming creates a new machine, because a general purpose computer in effect

result, software patents have circumvented the limits the 1952 Act places on functional claiming. The result has been a plethora of software patents claimed not on the basis of the technology the patentee actually developed, but on the basis of the function that technology performs. Those claims aren't limited to or commensurate with what the patentee invented, and they are accordingly the ones that patent plaintiffs tend to assert against defendants whose systems bear little resemblance to what the patentee actually invented.[90] And as Christina Bohannan and Herbert Hovenkamp note, under this functional claiming rubric the software patents with the least actual technical content end up with the broadest claims: "Its monopoly breadth is a function of its *lack* of technical specification."[91]

III. FUNCTIONAL CLAIMS AND THE TROUBLE WITH SOFTWARE PATENTS

A. *The Problem with Software Patents*

Software patents are widely acknowledged as creating a large number of problems for the patent system. Part of the problem is that there are so many software patents out there. Estimates vary widely, in part because it's hard to know what a software patent is, but there are certainly hundreds of thousands of software patents in force.[92] Because computer products tend to involve complex, multicomponent technology, any given product is potentially subject to a large number of patents. A few examples: 3G wireless technology was subject to more than 7,000 claimed "essential" patents as of 2004; the number is doubtless much higher now.[93] WiFi is subject to hundreds and probably thousands of

---

> becomes a special purpose computer once it is programmed to perform particular functions pursuant to instructions from program software.

Id. at 1545. By concluding that a general-purpose computer was a new machine whenever it was programmed with new instructions, the Federal Circuit opened the door to treating a programmed computer as physical structure rather than as a functional claim that had to be interpreted under Section 112(f).

90. *See* Kevin Emerson Collins, *Patent Law's Functionality Malfunction and Its Implications for the Problem of Overbroad, Functional Software Patents* 14–17 (Wash. Univ. Sch. of Law Legal Studies Research Paper Series, Paper No. 13-2-1, 2013), *available at* http://papers.ssrn.com/sol3/papers.cfm?abstract_id=2221950.

91. CHRISTINA BOHANNAN & HERBERT HOVENKAMP, CREATION WITHOUT RESTRAINT: PROMOTING LIBERTY AND RIVALRY IN INNOVATION 125 (2012).

92. For discussion and sources, see Mark A. Lemley & Carl Shapiro, *Patent Holdup and Royalty Stacking*, 85 TEX. L. REV. 1991, 2028–29 (2007).

93. *Id.* at 2025–26. Information on patents essential to 3G wireless technology is collected at http://www.3gpp2.org/, though that includes only patents disclosed to that

claimed essential patents.[94] And the problem is even worse than these numbers suggest, since both 3G wireless technology and WiFi are not themselves products but merely components that must be integrated into a final product. Some industry experts have estimated that 250,000 patents go into a modern smartphone.[95] Even nominally open-source technologies may turn out to be subject to hundreds or thousands of patents.[96] The result is what Carl Shapiro has called a "patent thicket"—a complex of overlapping patent rights that simply involves too many rights to cut through.[97]

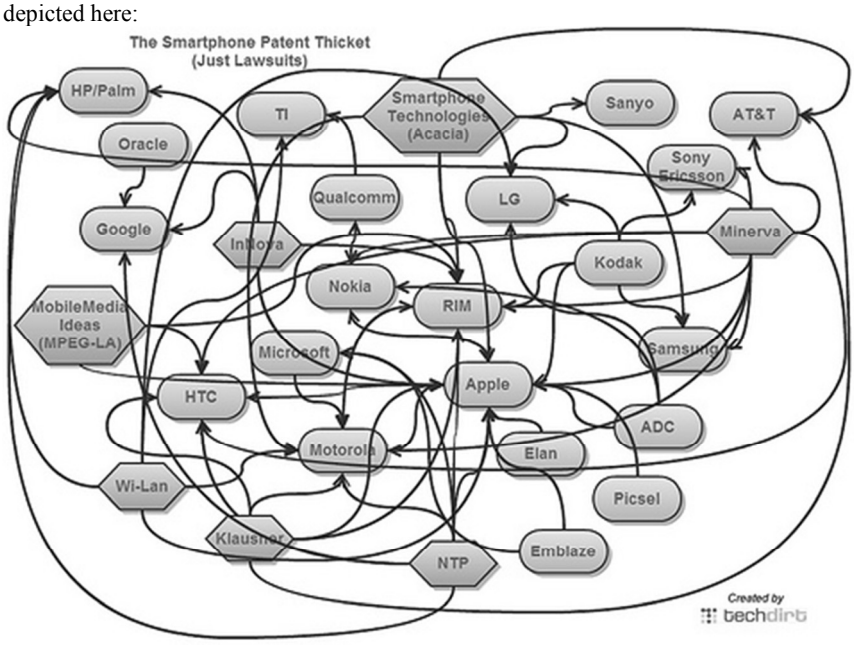
---

group. 3RD GENERATION PARTNERSHIP PROJECT 2, http://www.3gpp2.org/ (last visited Sept. 18, 2013).

94. *See* Ed Sutherland, *WiMax, 802.11n Renew Patent Debate*, WIFI PLANET (Apr. 7, 2005), http://www.wi-fiplanet.com/columns/article.php/3495951.

95. David Drummond, *When Patents Attack Android*, GOOGLE: OFFICIAL BLOG (Aug. 3, 2011), http://googleblog.blogspot.com/2011/08/when-patents-attack-android.html (statement of David Drummond, Chief Legal Officer at Google).

96. *Id*. (discussing patents that threaten the open-source Android operating system). Microsoft, Nokia, Apple, and others have all filed suit against makers of Android phones, part of a crazy tangle of litigation. The full panoply of lawsuits is depicted here:



http://www.flickr.com/photos/floorsixtyfour/5061246255/.

97. Carl Shapiro, *Navigating the Patent Thicket: Cross Licenses, Patent Pools, and Standard Setting*, in 1 INNOVATION POL'Y AND ECON. 119, 120 (2000). *See also* Michael A. Heller & Rebecca S. Eisenberg, *Can Patents Deter Innovation? The Anticommons in Biomedical Research*, 280 SCI. 698 (1998).

A related problem is the uncertainty associated with the meaning and scope of a software patent. Unlike chemistry and biotechnology, where we have a clear scientific language for delineating what a patent claim does and doesn't cover, there is no standard language for software patents. Accordingly, no one can really know what a software patent covers until the court has construed the language of the patent claims.[98] And because the Federal Circuit reverses as many as 40 percent of claim constructions,[99] the parties really can't know what a software patent covers until the Federal Circuit has addressed the issue. Compounding this problem, software patents in the 1980s and 1990s had to be disguised as something else in order to be patentable subject matter, which means that many early software patent claims were written to obfuscate what was in fact inventive about the technology.[100] Even worse, patentees can often benefit from ambiguous patent claims by twisting the language of the patent claim to cover something the inventor never in fact had in mind at the time.[101] Indeed, because computer technology changes so quickly, and it takes four years to get a patent out of the PTO on average, software patents are almost always asserted against technology that is several product generations removed from the patentee's invention, compounding the problem of trying to understand

98. Burk & Lemley, *Fence Posts*, *supra* note 15, at 1744–45.

99. For empirical studies of the high reversal rate in *Markman* hearings, see, for example, Christian A. Chu, *Empirical Analysis of the Federal Circuit's Claim Construction Trends*, 16 BERKELEY TECH. L.J. 1075 (2001); Kimberly A. Moore, *Are District Court Judges Equipped to Resolve Patent Cases?*, 15 HARV. J.L. & TECH. 1, 2–4 (2001); Kimberly A. Moore, Markman *Eight Years Later: Is Claim Construction More Predictable?*, 9 LEWIS & CLARK L. REV. 231, 231–34 (2005); David L. Schwartz, *Practice Makes Perfect? An Empirical Study of Claim Construction Reversal Rates in Patent Cases*, 107 MICH. L. REV. 223, 248–49 (2008). *Cf.* David L. Schwartz, *Pre*-Markman *Reversal Rates*, 43 LOY. L.A. L. REV. 1073 (2010) (studying reversal rates before district judges began expressly construing patent claims). Jonas Anderson and Peter Menell have found in a more recent study that the claim construction reversal rate is declining, but it is still over 25%. J. Jonas Anderson & Peter S. Menell, *Informal Deference: An Historical, Empirical, and Normative Analysis of Patent Claim Construction*, 108 NW. U. L. REV. (forthcoming 2012) (manuscript at 54), *available at* http://papers.ssrn.com/sol3/papers.cfm?abstract_id=2150360.

100. *See, e.g.*, Julie E. Cohen & Mark A. Lemley, *Patent Scope and Innovation in the Software Industry*, 89 CALIF. L. REV. 1, 3–5, 7–9 (2001) (discussing this history). Julie Cohen and I refer to the cases permitting software patents only if they pretended to be something else as establishing "the doctrine of the magic words." *Id.* at 9.

101. Burk & Lemley, *Fence Posts*, *supra* note 15, at 1762 ("In case after case, patentees claim to have invented electronic commerce, or multimedia, or video on demand, or voice-over-Internet, or call centers, or any of a hundred other successful technologies.").

the scope of software patents. The uncertainty problem is so bad that no one can agree on what a software patent even is.[102]

It is not just that the scope and definition of software patents are uncertain. Patents are probabilistic rights—what Carl Shapiro calls rights to *try* to exclude.[103] Many asserted software patents are invalid. Empirical evidence suggests that nearly half of all asserted patents are invalid;[104] there is some reason to believe software patents may be more likely than most to be invalid.[105] That means that even if a product-producing company could actually identify all of the thousands of patents that might ultimately be held to read on that product, they would be wasting their money in many cases if they tried to pay a license fee for each of those patents.

Among product-producing companies, the number and uncertainty of patents has created a patent "arms race" in which companies jockey to obtain more and more patents not in order to enforce those patents, but to protect themselves against the risk that competitors will enforce *their* patents.[106] The cost of this arms race can be staggering; in the last few

---

102. For various efforts to define software patents, see, for example, John R. Allison & Ronald J. Mann, *The Disputed Quality of Software Patents*, 85 WASH. U. L. REV. 297, 304–08, 313–15 (2007); Robert Hunt & James E. Bessen, *The Software Patent Experiment*, BUS. REV. (Fed. Reserve Bank of Phila.) 22, 24–26, 31 (2004), *available at* http://www.phil.frb.org/files/br/brq304rh.pdf; Arti K. Rai, John R. Allison & Bhaven N. Sampat, *University Software Ownership and Litigation: A First Examination*, 87 N.C. L. REV. 1519, 1529–33 (2009); James E. Bessen, *A Generation of Software Patents* 2–3, 12–14 (Boston Univ. Sch. of Law, Working Paper No. 11-31, 2011), *available at* http://ssrn.com/abstract=1868979; James E. Bessen & Robert M. Hunt, *An Empirical Look at Software Patents* 4–5 (Fed. Reserve Bank of Phila. Working Paper No. 03-17/R, 2004), *available at* http://www.researchoninnovation.org/swpat.pdf [hereinafter Bessen & Hunt, *Empircal Look*]. One evaluation found that these different methods of defining software patents had less than 30% overlap. *Id.* at 11 (advocating one measure of software patents, but finding that other measures differed from theirs as much as 74% of the time). That is, even the experts cannot agree most of the time on whether a patent even is a software patent.

103. *See* Mark A. Lemley & Carl Shapiro, *Probabilistic Patents*, 19 J. ECON. PERSP. 75, 93–95 (2005).

104. John R. Allison & Mark A. Lemley, *Empirical Evidence on the Validity of Litigated Patents*, 26 AIPLA Q.J. 185, 205 (1998) (Forty-six percent of litigated patent claims invalid).

105. *See* Allison et al., *Patent Quality*, *supra* note 12, at 707–09; Bessen & Hunt, *Empirical Look*, *supra* note 102, at 3–6; *but cf.* Allison & Mann, *supra* note 102, at 315–17, 333–34 (noting that the objective characteristics of software patents suggest that they are of high private value). High private value does not necessarily translate into validity; Allison et al. found that the most-litigated patents were extremely valuable even though most turned out to be invalid. Allison et al., *Patent Quality*, *supra* note 12, at 680.

106. *See* Colleen V. Chien, *From Arms Race to Marketplace: The Complex Patent Ecosystem and Its Implications for the Patent System*, 62 HASTINGS L.J. 297, 301, 339 (2010) [hereinafter Chien, *Arms Race*]; Colleen V. Chien, *Of Trolls, Davids,*

years companies in the smartphone industry have spent $15–20 billion buying patents to use in defending themselves against each other, and probably $1 billion just paying their lawyers.[107] And small companies must play the game too; by 2002 the overwhelming majority of software startups found it necessary to obtain patents even before going public[108]—which, given the four-year delay in the PTO, means that they must have started filing patent applications early indeed.

Spending billions of dollars to buy your own patents is not enough to protect an innovative software company from software patents. Patent "trolls"—those who don't practice their patented technology but sue others who do[109]—are legion in the software industry. Software and Internet patents are nearly ten times as likely to be enforced in court as other types of patents.[110] Empirical evidence suggests that the most-litigated patents (a group responsible for more than 10 percent of all patent assertions) are overwhelmingly software patents,

---

Goliaths, and Kings: Narratives and Evidence in the Litigation of High-Tech Patents, 87 N.C. L. REV. 1571, 1582, 1607 (2009) [hereinafter Chien, *Of Trolls*]. For a theoretical account of multi-patent portfolios, see Gideon Parchomovsky & R. Polk Wagner, *Patent Portfolios*, 154 U. PA. L. REV. 1 (2005).

107. Google bought Motorola Mobility for $12.5 billion. *Google to Acquire Motorola Mobility: Combination Will Supercharge Android, Enhance Competition, and Offer Wonderful User Experiences* (Aug. 15, 2011), http://investor.google.com/releases/2011/0815.html. A consortium of technology companies purchased Nortel's patent portfolio for $4.5 billion. Press Release, Research in Motion (RIM), RIM Participates in Winning Bid for Nortel's Patent Portfolio (July 1, 2011), press.blackberry.com/press/2011/pressrelease-5098.html. Microsoft bought some patents from AOL, and an exclusive license for other patents, in a deal worth over $1 billion. Press Release, AOL Inc., AOL and Microsoft Announce $1.056 Billion Patent Deal (Apr. 9, 2012), corp.aol.com/2012/04/09/aol-and-microscoft-announce-1-056-billion-patent-deal/. That is $18 billion, and does not include a number of smaller transactions under $500 million. Even if we credit $6 billion of the Motorola purchase to its hardware market, that's still $12 billion just for reported smartphone patent purchases; there are surely more that are confidential. In addition, my estimate based on conversations with people close to the cases is that the parties in the ongoing smartphone litigation have already spent at least $1 billion in legal fees, and the cases are far from over.

108. Rosemarie H. Ziedonis, *On the Apparent Failure of Patents: A Response to Bessen and Meurer*, 22 ACAD. MGMT. PERSP., 21, 26 fig.2. (2008). Ziedonis sees this as evidence that startups benefited from software patents, but it seems more likely evidence that they were caught up in the patent arms race. And Colleen Chien has shown that most patent troll suits are brought against small, not large, companies. Colleen V. Chien, *Startups and Patent Trolls*, 16 STAN. TECH. L. REV. (forthcoming 2013) (manuscript at 1), available at http://papers.ssrn.com/sol3/papers.cfm?abstract_id=2146251 [hereinafter Chien, *Startups*].

109. *See* Allison et al., *Patent Quality*, *supra* note 12, at 683.

110. John R. Allison et al., *Patent Litigation and the Internet*, 2012 STAN. TECH. L. REV. 3, 4 (2012), http://stlr.stanford.edu/pdf/allison-patent-litigation.pdf [hereinafter Allison et al., *Internet*].

overwhelmingly filed by patent trolls, and overwhelmingly unsuccessful in court.[111] Only about 10 percent of software patents in the most-asserted group actually prevail when the case goes to judgment.[112] Nonetheless, patent trolls are big business, representing more than half of all patent lawsuits in 2012 and an even higher percentage in the software industry.[113] They are mutating in form, with companies developing into "patent aggregators" that collect tens of thousands of patents and demand royalties to license the portfolio, suing those who don't pay.[114] And they have more recently been joined by "patent privateers"—product-producing companies who spin off patents or ally with trolls to target a competitor with lawsuits.[115] The result? According to one

111. *See* Allison et al., *Extreme Value or Trolls on Top? The Characteristics of the Most-Litigated Patents*, 158 U. PA. L. REV. 1 (2009) [hereinafter Allison et al., *Extreme Value*].

112. *Id*. at 686–89 ("[T]he most-litigated—and putatively most valuable—patents win in court only 10.7% of the time."); *see also* Allison et al., *Internet*, *supra* note 110, at 4 (stating win rate of Internet patents was extremely low).

113. *Cf.* Allison et al., *Internet*, *supra* note 110, at 4 (finding small entities were much more likely than large entities to enforce Internet patents). Credible estimates of the extent of patent troll litigation are hard to come by. Colleen Chien found several years ago that trolls filed 19% of all patent suits and targeted 36% of all defendants. Chien, *Of Trolls*, *supra* note 106, at 1604. But that number surely understates the role of trolls in the software industry because trolls are most prevalent in high-tech industries (and virtually unheard of in industries like pharmaceuticals). *See* Allison et al., *Extreme Value*, *supra* note 111, at 3 (noting that trolls own most of the most-litigated patents); Allison et al., *Patent Quality*, *supra* note 12, at 700–02 (finding that the most-litigated cases name substantially more defendants); James C. Pistorino & Susan J. Crane, *2011 Trends in Patent Case Filings: Eastern District of Texas Continues to Lead until American Invents Act Is Signed*, 83 PAT., TRADEMARK, & COPYRIGHT J. (BNA) 710 (2012) (suits filed in the Eastern District of Texas named many more defendants per case than suits elsewhere).

More recently, Lex Machina has found that 40% of the suits filed in 2011 were by "patent assertion entities"—companies primarily in the business of bringing patent suits. LEX MACHINA, http://www.lexmachina.com (last visited Sept. 8, 2013) (on file with author). Colleen Chien has found a dramatic increase in the number of patent troll suits, to 61% of all cases *See* Colleen V. Chien, *Presentation: Patent Assertion Entities*, 23 (Dec. 10, 2012), *available at* http://papers.ssrn.com/sol3/papers.cfm?abstract_id=2187314. Trolls are also behind the claims against 50% of the defendants in the International Trade Commission (ITC), even though the ITC nominally has a requirement that there be a domestic industry protected by the patent. *See* Colleen V. Chien & Mark A. Lemley, *Patent Holdup, the ITC, and the Public Interest*, 98 CORNELL L. REV. 1, 17 (2012). Most recently, Chien and Karkhanis found that 82% of software industry suits are brought by trolls, compared with 30% of non-software suits. Chien & Karkhanis, *supra* note 65, at 7.

114. *See* Tom Ewing & Robin Feldman, *The Giants among Us*, 2012 STAN. TECH. L. REV. 1, 1 (2012), http://stlr.stanford.edu/pdf/feldman-giants-among-us.pdf (documenting the behavior of one such patent aggregator, Intellectual Ventures).

115. *See* Tom Ewing, *Indirect Exploitation of Intellectual Property Rights by Corporations and Investors*, 4 HASTINGS SCI. & TECH. L.J. 1, 5 (2012).

estimate, trolls cost the economy $500 billion over the last twenty years, mostly in the information technology industry.[116] Other reports suggest that patent trolls inhibit innovation at the firms they sue.[117]

The combination of a thicket of hundreds of thousands of patents, the prevalence of patent trolls and their kin, the invalidity of many of those patents, and uncertainty as to what the patents actually cover means that companies in the software industry largely ignore patents unless and until they are threatened with suit.[118] But if a software product is successful, its maker can expect to be hit with dozens of suits and hundreds of threat letters from patent owners who come out of the woodwork and seek a royalty from that product.[119] Until recently, each of those patentees could credibly threaten to shut down the defendant's product altogether, even if the patent covered only a small fraction of the product. Even after the Supreme Court's decision in *eBay, Incorporated v. MercExchange LLC*[120] reduced the risk of injunction-related holdup,[121]

116. James E. Bessen et al., *The Private and Social Costs of Patent Trolls* 17 (Boston Univ. Sch. of Law, Working Paper No. 11-45, 2011), *available at* http://papers.ssrn.com/sol3/papers.cfm?abstract_id=1930272. While a functioning patent market might be a desirable thing, see Robert P. Merges, *The Trouble with Trolls: Innovation, Rent-Seeking, and Patent Law Reform*, 24 BERKELEY TECH. L.J. 1583, 1599 (2009), there is no reason to believe patent trolls are in fact engaged in much legitimate technology transfer.

117. Catherine Tucker, *Patent Trolls and Technology Diffusion* 4 (Tilburg Law & Econ. Ctr., Discussion Paper No. 2012-030), *available at* http://papers.ssrn.com/sol3/papers.cfm?abstract_id=2136955.

118. For evidence and discussion, see Mark A. Lemley, *Ignoring Patents*, 2008 MICH. ST. L. REV. 19, 21–22. *See also* Rebecca S. Eisenberg, *Patent Costs and Unlicensed Use of Patented Inventions*, 78 U. CHI. L. REV. 53, 54 (2011) (arguing that ignoring patents gives some freedom to technology companies to operate, but that they would be unwise to rely too heavily on forbearance by patent owners).

119. To take just a few examples, Lex Machina data shows that as of May 1, 2012, Apple had been named in 298 patent lawsuits over the last dozen years, Microsoft in 269 patent lawsuits, Google in 151, Yahoo! in 91, Oracle in 58, Facebook in 56, SAP in 38, Yelp in 9, and Twitter in 8. LEX MACHINA, http://www.lexmachina.com (last visited Sept. 8, 2013) (on file with author). While some of these companies, notably Apple and Oracle, are plaintiffs in some suits, the overwhelming majority of these cases involve the named companies as patent infringement defendants, and the majority are filed by patent trolls. Lemley et al. show in forthcoming work that filing an IPO attracts an average of eight patent lawsuits. Mark A. Lemley, Ziv Shafir, & Durgesh Saraph, "Because That's Where the Money Is": IPOs and Patent Suits (vaporware 2013) (unpublished manuscript) (on file with author).

120. 547 U.S. 388 (2006).

121. *Id.* at 394. eBay didn't eliminate the injunction-based holdup problem, however, even for suits by patent trolls. Trolls increasingly have turned to the ITC, an administrative agency that has the authority to exclude infringing products from entering the United States. And the ITC is not subject to *eBay Inc.*'s limits on injunctive relief. For discussion of the increasing use of the ITC by trolls, and what might be done about it, see

the fact that patentees have been able to seek large damage awards disproportionate to the value of the patented technology has created a "royalty stacking" problem.[122]

Software patents, then, have created a large number of problems for the industry, particularly for the most innovative and productive companies.[123] At the same time, software patents are arguably less necessary to spur innovation than are patents in other industries, such as pharmaceuticals or biotechnology. Software innovation is less costly than innovation in the life sciences.[124] Copyright also protects software and prevents copying by others.[125] Network effects may allow innovators to capture significant returns even absent IP protection.[126] And the existence of a vibrant open source community suggests that innovation can flourish in software absent patent protection.[127] If Michael Abramowicz and John Duffy are correct that we should only grant

---

Colleen V. Chien & Mark A. Lemley, *Patent Holdup, the ITC, and the Public Interest*, 98 CORNELL L. REV. 1 (2012); Colleen V. Chien, *Protecting Domestic Industries at the ITC*, 28 SANTA CLARA COMP. & HIGH TECH. L.J. 169 (2011).

122. Lemley & Shapiro, *supra* note 92, at 1993–94.

123. *See* Chien, *Startups*, *supra* note 108, at 1 (noting that trolls mostly target small companies).

124. DAN L. BURK & MARK A. LEMLEY, THE PATENT CRISIS AND HOW THE COURTS CAN SOLVE IT 38–41 (2009) [hereinafter BURK & LEMLEY, PATENT CRISIS] (citing evidence on relative cost of development).

125. 17 U.S.C. § 102(a) (2006).

126. *See, e.g.*, CARL SHAPIRO & HAL R. VARIAN, INFORMATION RULES: A STRATEGIC GUIDE TO THE NETWORK ECONOMY, NETWORK RULES 199–200 (1999); Mark A. Lemley & David McGowan, *Legal Implications of Network Economic Effects*, 86 CAL. L. REV. 479, 491–92 (1998); s*ee also* Joseph Farrell & Garth Saloner, *Standardization, Compatibility, and Innovation*, 16 RAND J. ECON. 70, 70 (1985); Michael L. Katz & Carl Shapiro, *Network Externalities, Competition, and Compatibility*, 75 AM. ECON. REV. 424, 424 (1985).

127. For discussion of the significance of open source software, see, for example, YOCHAI BENKLER, THE WEALTH OF NETWORKS 63–67 (2006); ERIC S. RAYMOND, THE CATHEDRAL AND THE BAZAAR: MUSINGS ON LINUX AND OPEN SOURCE BY AN ACCIDENTAL REVOLUTIONARY 64–67 (1999); Josh Lerner & Jean Tirole, *Some Simple Economics of Open Source*, 50 J. INDUS. ECON. 197, 212–15 (2002) (explaining the benefits programmers receive from participating in open source software development). On open source and its implications for law, see, for example, Yochai Benkler, *Coase's Penguin, or, Linux and* The Nature of the Firm, 112 YALE L.J. 369, 445–46 (2002); James E. Bessen, *Open Source Software: Free Provision of Complex Public Goods*, *in* THE ECONOMICS OF OPEN SOURCE SOFTWARE DEVELOPMENT 57, 79–80 (Jürgen Bitzer & Philipp J. H. Schröeder, eds., 2006); Michele Boldrin & David K. Levine, *Market Structure and Property Rights in Open Source Industries*, 30 WASH. U. J.L. & POL'Y 325, (2009); David McGowan, *Legal Implications of Open-Source Software*, 2001 U. ILL. L. REV. 241; Greg R. Vetter, *Commercial Free and Open Source Software: Knowledge Production, Hybrid Appropriability, and Patents*, 77 FORDHAM L. REV. 2087, 2129–31 (2009).

patents that encourage innovation we wouldn't have gotten otherwise,[128] software seems a poor candidate for patent protection.

The result has been that economic evidence suggests software patents impose significant costs on society. Jim Bessen and Mike Meurer estimate the social cost of patent trolls at an aggregate of $500 billion.[129] Elsewhere, the same authors find that patents in the information technology industry have a net negative effect on market value of companies in the industry.[130] While I have suggested elsewhere that trolls are a symptom of the problem, not the problem itself,[131] they are clearly evidence of a software patent system that has real problems.

B. Proposals to Reform Software Patents

In response to these problems, commentators have proposed several solutions to the problem of software patents.

1. ABOLISHING SOFTWARE PATENTS

A number of commentators have called for the abolition of software patents.[132] Dan Burk and I have argued elsewhere that such a remedy is

---

128. Michael Abramowicz & John F. Duffy, *The Inducement Standard of Patentability*, 120 YALE L.J. 1590, 1597–98 (2011). They are surely right as an abstract economic matter, though the idea seems impossible to implement in practice. *But cf.* Jonathan M. Barnett, *Intellectual Property as a Law of Organization*, 84 S. CAL. L. REV. 785, 808 (2011) (arguing that companies will substitute for the absence of patents with potentially inefficient organizational changes); Dan L. Burk & Brett H. McDonnell, *The Goldilocks Hypothesis: Balancing Intellectual Property Rights at the Boundary of the Firm*, 2007 U. ILL. L. REV. 575, 617.

129. Bessen et al., *supra* note 116, at 20, 32 tbl.3 (finding that little of this money is a transfer to patent trolls; most is a pure welfare loss).

130. JAMES E. BESSEN & MICHAEL MEURER, PATENT FAILURE: HOW JUDGES, BUREAUCRATS, AND LAWYERS PUT INNOVATORS AT RISK 137 (2008).

131. *See* Mark A. Lemley & A. Douglas Melamed, *Missing the Forest for the Trolls*, 113 COLUM. L. REV. (forthcoming 2013), *available at* http://papers.ssrn.com/sol3/papers.cfm?abstract_id=2269087.

132. *See, e.g.*, League for Programming Freedom, *Software Patents: Is This the Future of Programming?*, DR. DOBB'S J., Nov. 1990, at 56; Brian J. Love, *Why Patentable Subject Matter Matters for Software*, 81 GEO. WASH. L. REV. ARGUENDO 1 (2012), http://www.gwlr.org/wp-content/uploads/2012/09/Love_Arguendo_81_1.pdf (arguing that while Section 101 exclusion is problematic, it is "virtually the only defensive mechanism left"); Alan Newell, *Response: The Models Are Broken, The Models Are Broken!*, 47 U. PITT. L. REV. 1023, 1025 (1986); Pamela Samuelson, Benson *Revisited: The Case against Patent Protection for Algorithms and Other Computer Program-Related Inventions*, 39 EMORY L.J. 1025, 1135–36 (1990); Joshua D. Sarnoff, *Patent-Eligible Inventions after* Bilski*: History and Theory*, 63 HASTINGS L.J. 53,

overbroad; there are in fact real inventions in the software space that deserve patent protection.[133] And the line-drawing problems mentioned above mean that any effort to define a class of software exempt from patenting is doomed to be enmeshed in endless self-serving disputes over whether a particular invention is or isn't software.[134] Indeed, it may take us back to the bad old days of software patents that pretended to be something else. In any event, with hundreds of thousands of software patents issued over the past twenty-five years, it seems impractical to think Congress will simply ban software patents (though recent case law

---

119–20 (2011). *But see* Donald S. Chisum, *The Patentability of Algorithms*, 47 U. PITT. L. REV. 959, 1014–15 (1986); Mark A. Lemley et al., *Life after* Bilski, 63 STAN. L. REV. 1315, 1326–27 (2011) [hereinafter Lemley, et al., *Life after* Bilski]; Robert P. Merges, *Software and Patent Scope: A Report from the Middle Innings*, 85 TEX. L. REV. 1627, 1656–57 (2007); Michael Risch, *Everything Is Patentable*, 75 TENN. L. REV. 591, 622 (2008).

In *Bilski v. Kappos*, 130 S. Ct. 3218 (2010), four Justices would have drawn a similar line banning the patenting of business methods. *Id*. at 3232 (Stevens, J., concurring). *See also* Peter S. Menell, *Forty Years of Wandering in the Wilderness and No Closer to the Promised Land:* Bilski*'s Superficial Textualism and the Missed Opportunity to Return Patent Law to Its Technology Mooring*, 63 STAN. L. REV. 1289, 1312–13 (2011); John R. Thomas, *The Patenting of the Liberal Professions*, 40 B.C. L. REV. 1139, 1145–47 (1999).

133. BURK & LEMLEY, PATENT CRISIS, *supra* note 124, at 157–58. *See also* 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 198 (2004) (arguing against industry-specific patent rules).

134. BURK & LEMLEY, PATENT CRISIS, *supra* note 124, at 157–58; John F. Duffy, *Rules and Standards on the Forefront of Patentability*, 51 WM. & MARY L. REV. 609, 614 (2009) (stating that when it comes to patentable subject matter, "rules always fail"). Others have identified the particular difficulties courts and commentators have had in defining software patents. *See, e.g.*, Reinier B. Bakels, *Are Software Patents Something Special?*, *in* BIOTECHNOLOGY AND SOFTWARE PATENT LAW: A COMPARATIVE REVIEW OF NEW DEVELOPMENTS 131, 131–34 (Emanuela Arezzo & Gustavo Ghidini eds. 2011). To consider just one example of the line-drawing problem, take the Toyota Prius. Its hybrid gasoline-electric engine works because the car has a sophisticated controller that decides when to draw power from the gasoline engine and when from the battery. That controller is a piece of software. Is the hybrid car engine a "software patent"? *But see* John M. Golden, *Patentable Subject Matter and Institutional Choice*, 89 TEX. L. REV. 1041, 1111 (2011) (arguing for vesting significant power to limit patentable subject matter with the PTO).

John Allison and Starling Hunter have argued that the line-drawing problems are so great that trying to eliminate software patents would "prove largely futile and possibly even counterproductive—futile because skilled patent attorneys can often draft applications so as to opt out of a predefined category, and counterproductive because of the increased transaction costs associated with tortuous drafting." John R. Allison & Starling D. Hunter, *On the Feasibility of Improving Patent Quality One Technology at a Time: The Case of Business Methods*, 21 BERKELEY TECH. L.J. 729, 736 (2006); John R. Allison & Emerson H. Tiller, *The Business Method Patent Myth*, 18 BERKELEY TECH. L.J. 987, 1081–82 (2003).

on patentable subject matter may have a similar effect; more on that below).[135]

2. WEEDING OUT BAD PATENTS

Others have suggested that we can solve the problem by weeding out bad software patents, often by beefing up examination at the PTO, but sometimes by changing the legal standards so that courts are more likely to find a software patent obvious.[136] There is no question that there are bad software patents out there, and invalidating them is a social good.[137] But as I have argued elsewhere, it is not clear that we want to

---

135. *See infra* notes 230–32 and accompanying text.

136. *See, e.g.*, BESSEN & MEURER, *supra* note 130, at 247–48 (arguing for higher obviousness standards to reduce the flood of patents); JAFFE & LERNER, *supra* note 133, at 170–82 (arguing for efforts to improve patent quality); Julie E. Cohen, *Reverse Engineering and the Rise of Electronic Vigilantism: Intellectual Property Implications of "Lock-Out" Programs*, 68 S. CAL. L. REV. 1091, 1179 (1995); Robert P. Merges, *As Many as Six Impossible Patents before Breakfast: Property Rights for Business Concepts and Patent System Reform*, 14 BERKELEY TECH. L.J. 577, 589–90 (1999) (discussing the inadequacies of prior art searches in software); Andrew Nieh, *Software Wars: The Patent Menace*, 55 N.Y.L. SCH. L. REV. 295, 324–25 (2010–2011); Simson L. Garfinkel, *Patently Absurd*, WIRED, July 1994, at 104, 142. One commentator has described this approach as saying, not that software patents are bad, but that "bad software patents are bad." U.S. PATENT AND TRADEMARK OFFICE, PUBLIC HEARING ON USE OF THE PATENT SYSTEM TO PROTECT SOFTWARE-RELATED INVENTIONS 56 (Jan. 26 & 27, 1994) (statement of Ronald S. Laurie).

137. One study finds that at least 27% of all patents would be invalid if litigated. Shawn P. Miller, Where's the Innovation? An Analysis of the Quantity and Qualities of Anticipated and Obvious Patents 2 (Feb. 10, 2012) (unpublished manuscript) *available at* http://ssrn.com/abstract=2029263. In fact, that probably understates the problem. Nearly half of those patents actually litigated are held invalid. John R. Allison & Mark A. Lemley, *Empirical Evidence on the Validity of Litigated Patents*, 26 AIPLA Q.J. 185, 205 (1998). And while some have suggested that the high invalidity rate is a function of litigation selection effects, see George L. Priest & Benjamin Klein, *The Selection of Disputes for Litigation*, 13 J. LEGAL STUD. 1, 5 (1984), that argument is both theoretically unconvincing and empirically untrue in patent law. On the theory, see Daniel Kessler et al., *Explaining Deviations from the Fifty-Percent Rule: A Multimodal Approach to the Selection of Cases for Litigation*, 25 J. LEGAL STUD. 233 (1996); Steven Shavell, *Any Frequency of Plaintiff Victory at Trial is Possible*, 25 J. LEGAL STUD. 493, 498–501 (1996); Jason Rantanen, *Why Priest-Klein Cannot Apply to Individual Issues in Patent Cases* (U. Iowa Legal Studies Research Paper No. 12-15, 2013), *available at* http://papers.ssrn.com/sol3/papers.cfm?abstract_id=2132810. As for evidence, see Mark A. Lemley, *The Fractioning of Patent Law*, *in* INTELLECTUAL PROPERTY AND THE COMMON LAW 504, 506 n.14 (Cambridge University Press 2013) ("[E]very empirical study of patent law refutes it; each shows systematic variation from a 50% win rate.").

For arguments that litigants have insufficient incentive to challenge bad patents because invalidation of a patent benefits their competitors as well as themselves, see Joseph Farrell & Robert P. Merges, *Incentives to Challenge and Defend Patents: Why*

spend the money it would take to weed out every bad patent at the PTO because most of those patents have no ill effects.[138]

More important, while we could likely do better at weeding out bad patents in court, doing so would likely come at a cost, both in terms of legal fees and court time and in increasing the risk of wrongly invalidating legitimate patents.[139] In any event, weeding out bad patents in court would alleviate only some of the problems with software patents. While we wouldn't need to worry about erroneous injunctions or damage awards, companies would still face thousands of patents of uncertain validity and the need to pay millions of dollars in legal fees to invalidate each asserted patent.[140] And the software patents arms race has developed to such an extent that weeding out 50 percent, or even 90 percent, of software patents might still leave a significant thicket of broad patents with which innovators must contend. Smartphone companies, for instance, would likely take little solace in being told that they need only clear rights for 25,000 essential patents, not 250,000.

Finally, it is worth emphasizing that there are real technical inventions in software, just as in any other innovative area of technology. It is true that software patents today are invalidated more often than other types of patents, but that is a consequence of the remarkable breadth we have given those patent claims by allowing functional claiming. Many of those patents have at their heart real technical inventions.[141] Even if the law should treat almost all broad functional claims as obvious, that doesn't mean those inventors don't deserve a narrower patent commensurate with what they actually achieved.

Litigation Won't Reliably Fix Patent Office Errors and Why Administrative Patent Review Might Help, 19 BERKELEY TECH. L.J. 943, 958 (2004); Joseph Scott Miller, *Building a Better Bounty: Litigation-Stage Rewards for Defeating Patents*, 19 BERKELEY TECH. L.J. 667, 668–73 (2004); John R. Thomas, *Collusion and Collective Action in the Patent System: A Proposal for Patent Bounties*, 2001 U. ILL. L. REV. 305, 332–33.

138. Mark A. Lemley, *Rational Ignorance at the Patent Office*, 95 NW. U. L. REV. 1495, 1496–97 (2001). For a discussion of how the PTO might better target its resources on important patents, see Mark A. Lemley, *Fixing the Patent Office*, *in* 13 INNOVATION POLICY AND THE ECONOMY 83 (2013).

139. For an argument that most efforts to improve patent quality are likely to be ineffective or even counterproductive, see R. Polk Wagner, *Understanding Patent-Quality Mechanisms*, 157 U. PA. L. REV. 2135, 2163–65 (2009).

140. AIPLA, REPORT OF THE ECONOMIC SURVEY 2013, at 34 (reporting that a high-stakes patent case costs a median of $3 million per side in legal fees if it settles after discovery and $5.5 million if the case goes to trial).

141. *See* Jeanne C. Fromer, *The Layers of Obviousness in Patent Law*, 22 HARV. J.L. & TECH. 75, 96 (2008) (noting that implementing an idea in software is complex even once the idea is known) [hereinafter Fromer, *Layers*].

3. DEFINING THE SCOPE OF SOFTWARE PATENTS

Still others, including Bessen and Meurer, have suggested that the problem is the vagueness in the boundaries of software patents.[142] They argue that if we were clearer in indicating what software patents actually covered, people would be able to tell in advance which patents they needed to license.[143] As Bessen and Meurer put it, "if you can't tell the boundaries, then it ain't property."[144] They argue for a combination of limits on late claiming through patent "continuations" and a more robust effort to invalidate patents for "indefiniteness."[145]

Bessen and Meurer are surely correct that patents suffer from notice and boundary problems and that software patents suffer more than most.[146] They are also right to say that software patents are fundamentally unlike real property because the boundary disputes are so prevalent.[147] But it is unrealistic to think that we can somehow give software patents clear boundaries and make IP "like" real property. The problems are too fundamental.[148] They include:

- the process of peripheral claiming—trying to define a group of things (both known and as yet unknown) in words;[149]

---

142. BESSEN & MEURER, *supra* note 130, at 201–03.

143. *See id.* at 8–11.

144. *Id*. at 46. *See also* Peter S. Menell & Michael J. Meurer, *Notice Failure and Notice Externalities*, J. LEGAL ANALYSIS (forthcoming 2013), *available at* http://papers.ssrn.com/sol3/papers.cfm?abstract_id=1973171.

145. BESSEN & MEURER, *supra* note 130, at 25–26. On continuations and their problems, see Mark A. Lemley & Kimberly A. Moore, *Ending Abuse of Patent Continuations*, 84 B.U. L. REV. 63, 71–83 (2004). On definiteness, governed by 35 U.S.C. § 112(b), see BESSEN & MEURER, *supra* note 130, at 235–42.

146. *See* Love, *supra* note 132. Shawn Miller finds that the Federal Circuit is much more likely to reverse claim construction decisions involving software than other areas of technology, suggesting that the understanding of software claim boundaries is much less certain than in other fields. Shawn P. Miller, Do 'Fuzzy' Software Patent Boundaries Explain High Claim Construction Reversal Rates? 9 (Feb. 7, 2013) (working paper 2013), *available at* http://ssrn.com/abstract=2139146.

147. For an explanation of why patents are not "property" in any meaningful sense, see Mark A. Lemley, *Property, Intellectual Property, and Free Riding*, 83 TEX. L. REV. 1031 (2005) [hereinafter Lemley, *Free Riding*].

148. *See* Emily Michiko Morris, Res *or Rules? Patents and the (Uncertain) Rules of the Game*, 18 MICH. TELECOMM. & TECH. L. REV. 481, 494 (2012).

149. *Festo Corp. v. Shoketsu Kinzoku Kogyo Kabushiki Co.*, 535 U.S. 722, 731 (2002) ("This conversion of machine to words allows for unintended idea gaps which cannot be satisfactorily filled. Often the invention is novel and words do not exist to describe it. The dictionary does not always keep abreast of the inventor. It cannot. Things are not made for the sake of words, but words for things.") (quoting *Autogiro Co. of Am.*

- the process of claim construction, in which lawyers take words that have been substituted for technological concepts and replace them with other, theoretically clearer words;[150]

- the four-year average delay in issuing a patent;[151] imagine being unable to know for several years whether you were building on land you owned or not;

- the practice of continuation applications, which permit applicants to add to or change the scope of their claims at any time during the twenty-year patent term;[152] and

- the sheer number of patent rights covering software. Real property lawyers tend to deal with boundary disputes between two or three parties. In an extreme case, someone who wants to acquire a large parcel of land may have to deal with dozens of different landholders. They don't have to deal with 250,000 patents owned by perhaps a thousand different entities;[153]

Bessen and Meurer offer a number of useful suggestions to deal at the margins with some of the uncertainties of software patents.[154] But at

---

v. United States, 384 F.2d 391, 397 (Ct. Cl. 1967)). For discussion of the inherent indeterminacy of this process, see Burk & Lemley, *Fence Posts*, *supra* note 15, at 1748–61; Lefstin, *supra* note 22, at 1204–10.

150. Burk & Lemley, *Fence Posts*, *supra* note 15, at 1744–46. For discussion of the ensuing debates over the meaning of the words used to construe the original words of the claim—so-called "meta-construction"—see Kristen Osenga, *Linguistics and Patent Claim Construction*, 38 RUTGERS L.J. 61, 69–70 (2006).

151. Dennis Crouch, *Total Patent Application Pendency*, PATENTLY-O (Mar. 18, 2012, 9:43 AM), http://www.patentlyo.com/patent/2012/03/total-patent-application-pendency.html ("[The] average and median pendency is just under five years."). The number has increased over the past fifteen years; in 1998 the average delay was 2.77 years. John R. Allison & Mark A. Lemley, *Who's Patenting What? An Empirical Exploration of Patent Prosecution*, 53 VAND. L. REV. 2099, 2118 (2000).

152. Lemley & Moore, *supra* note 145, at 66–69.

153. Indeed, Christina Mulligan and Timothy Lee estimate that there are "around twenty-four billion new [software] patent-firm pairs each year that could produce accidental infringement," and that to hire a lawyer to spend even ten minutes reviewing each patent for infringement would require two million lawyers working full time on clearing software patent rights. Christina Mulligan & Timothy B. Lee, *Scaling the Patent System*, 68 N.Y.U. ANN. SURV. AM. L. 289, 304–05 (2012).

154. For a more systematic approach to improving notice, while recognizing that some uncertainty is inevitable, see generally Harry Surden, *Efficient Uncertainty in Patent Interpretation*, 68 WASH. & LEE L. REV. 1737 (2011).

the end of the day, the problem is that there are simply too many patents owned by too many people that claim to be essential to practicing modern computer technology. If there are too many patent rights that are too broad, making their boundaries clearer will only show us the magnitude of the problem we face; it won't solve that problem for us.

4. AN INDEPENDENT INVENTION DEFENSE

Another possibility is to change the rule that independent invention is not a defense to patent infringement. Unlike copyright and trade secret law, patent rights are enforceable against anyone who makes a product incorporating the patented invention, whether or not they got the idea from the patentee.[155] A number of scholars have proposed changing this rule.[156] Eliminating cases filed against independent inventors would have a major effect on the patent system because evidence suggests that independent inventors represent roughly 90 percent of those sued for patent infringement today.[157] And it would have a particular effect on patent trolls because they do not make products that can be copied in the marketplace and rarely engage in actual transfer of technology to product-producing companies.[158] But an independent invention defense works best if the patents in question are technical so that it is easy for a court to tell whether the accused infringer had a research trail that led to her developing the same idea independently. It would be much harder to tell whether functional claims are copied. For example, suppose Apple sues Samsung for implementing "swipe-to-unlock" functionality on its smartphones.[159] If the patent claim covers a particular algorithm, it should be straightforward to find out whether Samsung actually implemented that algorithm and, if so, what the process of developing it

155. Mark A. Lemley, *Should Patent Infringement Require Proof of Copying?*, 105 MICH. L. REV. 1525, 1525 (2007).

156. *E.g.*, Samson Vermont, *Independent Invention as a Defense to Patent Infringement*, 105 MICH. L. REV. 475 (2006) [hereinafter Vermont, *Independent Invention*]; Samson Vermont, *The Angel Is in the Big Picture: A Response to Lemley*, 105 MICH. L. REV. 1537 (2007); Carl Shapiro, *Prior User Rights*, 96 AM. ECON. REV. 92, 95 (2006).

157. *See* Christopher A. Cotropia & Mark A. Lemley, *Copying in Patent Law*, 87 N.C. L. REV. 1421, 1443 (2009).

158. For an argument that patentees should have to practice their products to be entitled to enforce them see, for example, Christopher A. Cotropia, *The Folly of Early Filing in Patent Law*, 61 HASTINGS L.J. 65 (2009); *see also* Ted Sichelman, *Commercializing Patents*, 62 STAN. L. REV. 341 (2010).

159. It has. Jury Verdict, *Apple, Inc. v. Samsung Elec., Co.* (N.D. Cal. Aug. 24, 2012) (No. 11-CV-01846-LHK).

looked like.[160] But if the patent claim covers the concept itself, figuring out whether someone at Samsung had the same basic idea or instead learned it by observing Apple's phone will be much harder. So whether or not an independent invention defense is a good idea in general,[161] it is hard to implement in a world of functional software claims.

IV. FUNCTIONAL CLAIMING AND THE SOFTWARE PATENT THICKET

None of the ideas I discussed in the last Part are likely to solve the problems we face with software patents. Some of the ideas are unrealistic, some come with unintended consequences, and all of them ignore a key element of the problem: the fact that we allow patentees to claim functions, not implementations. It is broad functional claiming that leads to assertions that every part of a complex technology product is patented, often by many different people at the same time. It is broad functional claiming that puts stars in the eyes of patent plaintiffs, who can demand huge royalties on the theory that there simply is no other way to implement the technology they have patented. And it is broad functional claiming that makes most of the resulting patents invalid, since even if ten programmers developed ten different algorithms to solve a problem, only one of them could be the first to solve the problem at all.

In this Part, I explain how a simple application of existing legal doctrine can end broad functional claiming of software. I also address objections and complications to treating functional software claims like other types of functional patent claims.

A. Taking Section 112(f) Seriously

Fortunately, there is no need to rewrite the patent law or retroactively invalidate tens of thousands of software patents in order to address the problem of functional claiming. All we need to do is take seriously the law already on the books.

While we refer to functional claiming under Section 112(f) as "means-plus-function" claiming, after a common language format ("means for doing x") that has been held to invoke that section, what the statute actually says is instructive:

---

160. *See* Vermont, *Independent Invention*, *supra* note 156 (arguing for the adoption of an independent invention defense to patent infringement, and suggesting that courts are a good forum for resolving such disputes).

161. I've expressed some concerns elsewhere. Lemley, *supra* note 155, at 1527–32.

> An element in a claim for a combination may be expressed as a means or step for performing a specified function without the recital of structure, material, or acts in support thereof, and such claim shall be construed to cover the corresponding structure, material, or acts described in the specification and equivalents thereof.[162]

The question in applying Section 112(f), then, is not whether the language is written in the form "means for doing x." It is whether a particular claim element is expressed "as a means or step for performing a specified function without the recital of structure, material, or acts in support thereof." If so, the second phrase of Section 112(f) applies, and the claim is to be construed by reference to the specification.

The Federal Circuit has said that use of the term "means" creates a presumption that a claim element is a means-plus-function element to which Section 112(f) applies, and the absence of that term creates the opposite presumption.[163] But the presumption can be rebutted either by evidence that the element in question isn't functional[164] or that the claim element contains a sufficiently "definite structure" to avoid invoking the statute.[165] At least in theory, then, deciding whether to turn to the specification to limit an allegedly means-plus-function claim element requires some inquiry into the claim language and whether it would be understood by scientists in the field to recite known structure.[166] If it does, the structure itself is a limitation, and there is no need to turn to the patent specification to find that limitation.[167]

In practice, however, the software cases draw a pretty formalistic line between claims that use the "means for doing x" language and those that don't.[168] On the one hand, when software patents are actually written

---

162. 35 U.S.C. § 112(f) (2012).

163. *See, e.g.*, *York Prods., Inc. v. Central Tractor Farm & Family Ctr.*, 99 F.3d 1568, 1574 (Fed. Cir. 1996).

164. *See, e.g.*, *Rodime PLC v. Seagate Tech., Inc.*, 174 F.3d 1294, 1302 (Fed. Cir. 1999) ("[A] claim element that uses the word 'means' but recites no function corresponding to the means does not invoke" Section 112(f).).

165. *Cole v. Kimberly-Clark Corp.*, 102 F.3d 524, 531 (Fed. Cir. 1996).

166. *Watts v. XL Sys.*, 232 F.3d 877, 880–81 (Fed. Cir. 2000).

167. That inquiry can be difficult. *Compare Cole*, 102 F.3d at 531 (concluding that "perforation means for tearing" was not a means-plus-function limitation because perforations were the structure for accomplishing the tearing function), *with Unidynamics Corp. v. Automatic Prods. Int'l*, 157 F.3d 1311 (Fed. Cir. 1998) (concluding that "spring means tending to keep the door closed" was a means-plus-function limitation because the term "spring" was part of the function, not itself a definite structure).

168. For criticism that the Federal Circuit takes an excessively formalist view in its jurisprudence, preferring bright lines to more flexible standards, see, for example, BURK & LEMLEY, PATENT CRISIS, *supra* note 124, ch. 7; Timothy R. Holbrook, *The*

using "means for doing x" language, the Federal Circuit has been quite strict about requiring evidence of real computer programming in the specification. Software patents that use means-plus-function language but do not detail actual algorithms implementing those functional steps are invalid for indefiniteness.[169] And courts are willing to ignore linguistic games and focus on what is really at issue, treating an invention that occurs primarily in software as requiring disclosure of software algorithms, not just computer hardware.[170] *Dealertrack Incorporated v. Huber*[171] is instructive:

> A general purpose computer can perform the claimed function of "executing a computer program which implements and controls credit application processing and routing" only if the program it executes is capable of performing those functions. That the true functional requirements of the limitation are nested within the generic function of executing a program does not change this fact; though the computer itself may execute a computer program, it may not execute *that* computer program without the algorithms.[172]

Supreme Court's Complicity in Federal Circuit Formalism, 20 SANTA CLARA COMPUTER & HIGH TECH. L.J. 1, 1–2 (2003); Lucas S. Osborn, *Instrumentalism at the Federal Circuit*, 56 ST. LOUIS U. L.J. 419, 425–26 (2012); Arti K. Rai, *Engaging Facts and Policy: A Multi-Institutional Approach to Patent System Reform*, 103 COLUM. L. REV. 1035, 1102–22 (2003); John R. Thomas, *Formalism at the Federal Circuit*, 52 AM. U. L. REV. 771 (2003). For a more favorable view of the Federal Circuit's formalism, see Peter Lee, *Patent Law and the Two Cultures*, 120 YALE L.J. 2, 27–29 (2010).

169. *See, e.g.*, *Function Media, LLC v. Google Inc.*, 708 F.3d 1310, 1318–19 (Fed. Cir. 2013) (use of flowcharts not sufficient because they just further described function; they did not explain what software actually performed that function); *Ergo Licensing, LLC v. CareFusion 303, Inc.*, 673 F.3d 1361, 1364 (Fed. Cir. 2012); *HTC Corp. v. IPCom GmbH & Co., KG*, 667 F.3d 1270, 1280, 1282–83 (Fed. Cir. 2012); *In re Aoyama*, 656 F.3d 1293, 1294, 1297–98 (Fed. Cir. 2011); *see also Noah Sys., Inc. v. Intuit Inc.*, 675 F.3d 1302, 1318–19 (Fed. Cir. 2012) (stating that where a means-plus-function claim element claims two functions, the specification must disclose algorithms implementing both functions); *Aristocrat Techs. Austl. PTY Ltd. v. Int'l Game Tech.*, 521 F.3d 1328, 1337–38 (Fed. Cir. 2008). *Cf. Typhoon Touch Techs., Inc. v. Dell, Inc.*, 659 F.3d 1376, 1384–86 (Fed. Cir. 2011) (concluding that an algorithm necessary to serve as structure of a means-plus-function software claim element need not be detailed or written in the form of computer code). *HTC* is particularly notable because the court went out of its way to indicate that the absence of an algorithm was a problem even though the issue had not been raised by the parties and was accordingly waived. *HTC*, 667 F.3d at 1282.

170. *See, e.g.*, *Dealertrack, Inc. v. Huber*, 674 F.3d 1315, 1333–34 (Fed. Cir. 2012).

171. *Id.*

172. *Id*. at 1329.

This is exactly right. But take exactly the same functional claim language, and replace individual "means for doing x" steps with a generic reference to a general-purpose computer "programmed to" achieve those same steps, as in the claims detailed in Part II, and the Federal Circuit no longer treats the claim as a means-plus-function claim and accordingly puts no limit on the functional nature of the claim.[173] Indeed, parties no longer even think about whether there is structure in those claims. In short, current cases treat "a computer" (or equivalents like "a processor connected to a memory") as a structural definition of the software invention, except where the patentee happened to make the mistake of using the word "means" to refer to that computer.[174]

This distinction ignores the realities of modern computer technology. Software patents by definition require implementation in a computer. Indeed, the Federal Circuit has recognized in other contexts that a computer is implicit in a software patent even if it appears nowhere in the claims.[175] Adding a term that is both necessary for any possible implementation of the function and so general as to impose no limit on the scope of the claim does not fit with the purpose of Section 112(f). The goal of Section 112(f) was to limit functional claiming by tying it to particular structure disclosed in the specification. If patentees can simply add "structure" in the form of inherently necessary technology, the purpose of that section is lost. It is as though a patentee had added the phrase "man-made" to a patent claiming "means for flying" and pointed to that as a structural limitation sufficient to take his invention outside the scope of functional claiming.

The "structure, material or acts" that must support a claim in functional language must be more than mere window-dressing. The intent of this statute was to allow functional claiming only when it was limited to particular implementations of that function, not when it encompassed all feasible ways of achieving the goal.[176]

---

173. *See supra* notes 92–95 and accompanying text.

174. *Id*.

175. *In re Dossel*, 115 F.3d 942, 946–47 (Fed. Cir. 1997) (written description); *Robotic Vision Sys., Inc. v. View Eng'g, Inc.*, 112 F.3d 1163, 1166 (Fed. Cir. 1997) (best mode).

176 *See supra* Part I (discussing the enactment of Section 112(f)); *cf. Regents of the Univ. of Calif. v. Eli Lilly & Co.*, 119 F.3d 1559, 1568 (Fed. Cir. 1997) (rejecting patent claim because "[i]t is only a definition of a useful result rather than a definition of what achieves that result."); *Ariad Pharms. Inc. v. Eli Lilly & Co.*, 598 F.3d 1336, 1353 (Fed. Cir. 2010) (en banc) ("Such claims merely recite a description of the problem to be solved while claiming all solutions to it . . . ."). For a broader suggestion to apply Section 112(f) to all patent claims, see Patrick G. Burns, *A Simpler Approach to Claim Construction*, 77 PAT., TRADEMARK, & COPYRIGHT J. (BNA) 717 (2009).

Fortunately, the solution to the problem is correspondingly simple: we must take seriously the dictate of Section 112(f).[177] If we limit patent claims that purport to cover functions to the actual structure, material, or acts the patentee built or described, the result will be that software patents will cover, not every possible way of implementing a goal, but the way the patentee actually implemented the goal "and equivalents thereof." And in computer software, the "structure" or "acts" that perform the function are not simply "a computer" or "a client-server system" but "a computer programmed in a particular way." That is, the structure of a software patent must involve software, not just the hardware substrate on which all software runs. Specifically, as recent Federal Circuit indefiniteness cases have shown, patentees will have to disclose the algorithms they use to achieve particular ends, and the patent will be limited to those algorithms and equivalents thereof.[178] This will leave room for later entrants to design around the patent and develop different algorithms to achieve the same result.[179]

We don't need to change the statute to achieve this result. We don't even need to overrule existing cases. We just need to take seriously law that is on the books but doesn't seem to get applied in practice. The Federal Circuit or the Supreme Court could, with one fell swoop, do

177. The Board of Patent Appeals and Interferences took a step in this direction in *Ex Parte Rodriguez*, No. 2008-693 (B.P.A.I. Oct. 1, 2009). There, the Board defined as means-plus-function claim elements the phrases: "system configuration generator configured to generate," "system builder configured to build," and "simulation verification environment configured to verify." *Id.* at 20. The Board found that these terms had no common structural meaning and so they were properly understood as means-plus-function elements. These claim elements referred to computer technology, though they didn't use any terms that expressly connoted computer hardware. *Id.* at 23. It remains to be seen whether the Board or the Federal Circuit will apply this principle to other recitations of generic computer technology.

178. *See supra* note 169 and accompanying text.

179. *Warner-Jenkinson Co. v. Hilton Davis Chem. Co.*, 520 U.S. 17, 36 (1997) (contrasting "the intentional copyist making minor changes to lower the risk of legal action" with "the incremental innovator designing around the claims, yet seeking to capture as much as is permissible of the patented advance"); *see also Slimfold Mfg. Co. v. Kinkead Indus.*, 932 F.2d 1453, 1457 (Fed. Cir. 1991) ("Designing around patents is, in fact, one of the ways in which the patent system works to the advantage of the public in promoting progress in the useful arts, its constitutional purpose."); *State Indus. v. A.O. Smith Corp.*, 751 F.2d 1226, 1236 (Fed. Cir. 1985) ("One of the benefits of a patent system is its so-called 'negative incentive' to 'design around' a competitor's products, even when they are patented, thus bringing a steady flow of innovations to the marketplace."); Matthew J. Conigliaro et al., *Foreseeability in Patent Law*, 16 BERKELEY TECH. L.J. 1045, 1048 (2001); Craig Allen Nard, *A Theory of Claim Interpretation*, 14 HARV. J.L. & TECH. 1, 40–41 (2000) ("The practice of designing-around extant patents creates viable substitutes and advances, resulting in competition among patented technologies. The public clearly benefits from such activity.").

away with most of the problem of over-claiming in software patents—and with it, most of the problems with software patents. All it needs to do is to take the statute at face value and limit functional claims to the particular way the patentee implemented that function. In the software world, the way an inventor implements a function is not with "a computer" or "a processor" but with a particular computer program. The patent claim should accordingly be limited to that particular computer program and ones that work in the same way to achieve the same result.[180]

The fact that we don't need to change the statute to achieve this result has an important benefit. While changes to statutes generally operate prospectively, new court interpretations of existing statutes are normally retroactive.[181] The idea is that the law hasn't changed; we simply understand it better. Retroactivity is key to solving the software patent thicket; it wouldn't do much good to say that patents issued four years from now will be narrower if we are stuck with hundreds of thousands of overbroad patents in force for the next two decades.[182] If the courts refused to act, Congress might be able to prompt action with a modest change to Section 112(f), perhaps by adding a sentence that reads, "If the function of an element is performed by software, recital of the medium on which software is stored or performed, such as computer hardware, is not sufficient to avoid application of this subsection." But I emphasize that this is a problem courts have created, and that courts should be the ones who solve it. And any Congressional action should make clear that Congress does not intend to change the law, but rather to

180. *See* BURK & LEMLEY, PATENT CRISIS, *supra* note 124, chs. 6, 11 (arguing that software patents should be narrow to allow for cumulative innovation); Randall M. Whitmeyer, Comment, *A Plea for Due Processes: Defining the Proper Scope of Patent Protection for Computer Software*, 85 NW. U. L. REV. 1103, 1106 (1991) ("[I]n the computer software context only narrow algorithms, as the term is understood by computer scientists, should be patentable.").

181. *See, e.g.*, David L. Schwartz, *Back from the Future: Retroactivity at the Federal Circuit*, 89 IND. L.J. (forthcoming 2014) (manuscript at 43–45), *available at* http://papers.ssrn.com/sol3/papers.cfm?abstract_id=1945554 (discussing the circumstances under which patent decisions should be retroactive). In a larger sense, the prospective-retroactive distinction is illusory; all private actors necessarily make assessments about how the law might change in the future. *See* Michael J. Graetz, *Legal Transitions: The Case of Retroactivity in Income Tax Revision*, 126 U. PA. L. REV. 47, 49–50 (1977); Louis Kaplow, *An Economic Analysis of Legal Transitions*, 99 HARV. L. REV. 509, 511–12 (1986).

182. Some have argued that any effort to narrow patent rights is a taking under the Fifth Amendment. *See, e.g.*, J. Nicholas Bunch, *Takings, Judicial Takings, and Patent Law*, 83 TEX. L. REV. 1747, 1762 (2005). Even if that were true, were the law changed to eliminate patent rights, it surely is not true of a court decision that actually applies a statute that has been on the books for sixty years.

push courts to apply existing law as written so that the change should apply retroactively.

B. Objections

In this Section, I consider two sets of objections, one rooted in asking whether such a change will really accomplish very much, and the other asking whether it will unfairly disadvantage inventors of software patents.

1. WILL IT WORK?

Some might question whether taking Section 112(f) seriously as a solution to the problem of software patents will limit software patents sufficiently. There are two components to this worry. First, courts might treat software claims implemented in general-purpose computers as means-plus-function claims but still not limit those claims to an algorithm or other detailed invention structure. The Federal Circuit record on this point is mixed. For software claims the court recognizes as invoking Section 112(f), most decisions have required that the patent specification disclose an algorithm for performing the specified function.[183] But on occasion the Federal Circuit has been more lenient to patentees, permitting them to satisfy the "particular and definite structure" with fairly general language rather than a specific implementation.[184] That is particularly true when hardware is at stake. Indeed, in one recent case the court went out of its way to find that the phrase "system memory means" was a means-plus-function claim element because it lacked a "specific and definite structure," only to find that the structure disclosed in the patent that corresponded to this claim was . . . wait for it . . . "a system memory."[185] An interpretation of

183. *Function Media, LLC v. Google Inc.*, 708 F.3d 1310, 1318–19 (Fed. Cir. 2013); *Ergo Licensing, LLC v. CareFusion, Inc.*, 673 F.3d 1361, 1364 (Fed. Cir. 2012); *HTC Corp. v. IPCom GmbH & Co., KG*, 667 F.3d 1270, 1280, 1282–83 (Fed. Cir. 2012); *In re Aoyama*, 656 F.3d 1293, 1297–98 (Fed. Cir. 2011); *Aristocrat Techs. Austl. PTY Ltd. v. Int'l Game Tech*., 521 F.3d 1328, 1333, 1337–38 (Fed. Cir. 2008); *see also Noah Sys., Inc. v. Intuit Inc.*, 675 F.3d 1302, 1318–19 (Fed. Cir. 2012) (stating that where a means-plus-function claim element claims two functions, the specification must disclose algorithms implementing both functions).

184. *See Typhoon Touch Techs., Inc. v. Dell, Inc.*, 659 F.3d 1376, 1385–86 (Fed. Cir. 2011) (finding that an algorithm necessary to serve as structure of a means-plus-function software claim element need not be detailed or written in the form of computer code); *In re Katz Interactive Call Processing Patent Litig.*, 639 F.3d 1303, 1316 (Fed. Cir. 2011).

185. *Chi. Bd. Options Exch., Inc. v. Int'l Sec. Exch., LLC*, 677 F.3d 1361,

Section 112(f) that does nothing more than replace the broad functional language of the claim with identical broad functional language from the specification renders that statute worthless.[186]

Fortunately, the majority of decisions from the Federal Circuit have not taken such a feckless approach. While the court does not necessarily require the disclosure of actual computer code to support the functional steps of a software claim,[187] it has tended to limit those claims to particular actual implementations of the idea, not to generic recitations of the functions the program performs.[188] And the court has gone out of its way to reject efforts by patentees to support functional claim language with specification language that just describes the function in more detail.[189]

The majority's narrower approach is consistent with the Supreme Court's approach to the question in *Halliburton*. Contrast the breadth of treating "system memory" as the relevant structure with the holding in *Halliburton*. The Court there wanted evidence of how the device that performed the function was actually constructed and how it connected with the rest of the invention:

> Walker, in some of his claims, for example, claims 2 and 3, does describe the tuned acoustical pipe as an integral part of his invention, showing its structure, its working arrangement in the alleged new combination, and the manner of its connection with the other parts. But no one of the claims on which this judgment rests has even suggested the physical structure of the acoustical resonator. No one of these claims describes the physical relation of the Walker addition to the old Lehr and Wyatt machine. No one of these claims describes the manner in

1366–69 (Fed. Cir. 2012). *But see Ergo Licensing*, 673 F.3d at 1363–64 ("The recitation of 'control device' provides no more structure than the term 'control means' itself, rather it merely replaces the word 'means' with the generic term 'device.'").

186. To be clear, a "system memory" may have a sufficiently clear meaning—and be sufficiently peripheral to the claim—that there is no real value to limiting the patentee to particular types of memories. But it is important not to apply a similar generic approach to the novel algorithmic steps of the patent. *See generally* Mark A. Lemley, *Point of Novelty*, 105 NW. U. L. REV. 1253 (2011) [hereinafter Lemley, *Novelty*].

187. *Typhoon Touch*, 659 F.3d at 1385–86.

188. *See supra* notes 183–85 and accompanying text (collecting cases).

189. *ePlus, Inc. v. Lawson Software, Inc.*, 700 F.3d 509, 518–20 (Fed. Cir. 2012). *See also Signtech USA, Ltd. v. Vutek, Inc.*, 174 F.3d 1352, 1356 (Fed. Cir. 1999) ("Although patentees are not necessarily limited to their preferred embodiment, . . . interpretation of a means-plus-function element requires this court to consult the structure disclosed in the specification, which often, as in this case, describes little more than the preferred embodiment." (citations omitted).

> which the Walker addition will operate together with the old Lehr and Wyatt machine so as to make the 'new' unitary apparatus perform its designed function. Thus the claims failed adequately to depict the structure, mode, and operation of the parts in combination.[190]

For Section 112(f) to serve as a real limit on functional claiming of software, courts must resist the temptation to permit broad generic recitations of structure in a means-plus-function claim, at least at the point of novelty, and return instead to the animating idea behind the statutory limitation on functional claiming.

The second worry stems from the "and equivalents thereof" language of Section 112(f). Because means-plus-function claim elements are not limited strictly to the structure disclosed in the specification but can encompass equivalent structures, patentees will have an incentive to argue that all forms of computer implementation of an idea are "equivalent" and so are covered within the literal bounds of the patent claim. Obviously they are equivalent in function; the functions must be identical for literal infringement under Section 112(f). So the question is whether patentees can persuade courts to equate different algorithmic approaches to solving the problem. If patentees can recapture all possible means of performing the function in this way, they will be able to avoid any limitation imposed by the structure and effectively own a functional claim.

I am less concerned that equivalents will allow such recapture for two reasons. First, there is an important difference between equivalents under the doctrine of equivalents and equivalents under Section 112(f). Section 112(f) equivalents do not apply to later-developed structures, but only to equivalents known at the time the patent issued.[191] Because software changes so quickly, most litigated software patents today are

---

190. *Halliburton Oil Well Cementing Co v. Walker*, 329 U.S. 1, 8 (1946). In the 1952 Act, Congress did not expressly overrule *Halliburton*, but rather said it was superseded by the new rules in Section 112(f). *See Warner-Jenkinson Co. v. Hilton Davis Chem. Co.*, 520 U.S. 17, 27–28 (1997).

191. *See, e.g.*, *Al-Site Corp. v. VSI Int'l, Inc.*, 174 F.3d 1308 (Fed. Cir. 1999) ("An equivalent structure or act under § 112 cannot embrace technology developed after the issuance of the patent because *the literal meaning of a claim is fixed upon its issuance*.") (emphasis added); *Chiuminatta Concrete Concepts, Inc. v. Cardinal Indus., Inc.*, 145 F.3d 1303, 1310 (Fed. Cir. 1998). For a discussion of this timing question, see Mark A. Lemley, *The Changing Meaning of Patent Claim Terms*, 104 MICH. L. REV. 101, 107–08 (2005) [hereinafter Lemley, *Changing Meaning*]. By contrast, the doctrine of equivalents can encompass equivalent *functions* as opposed to structures. *See WMS Gaming, Inc. v. Int'l Game Tech.*, 184 F.3d 1339, 1353 (Fed. Cir. 1999).

asserted against technologies that did not exist at the time of patenting.[192] This is especially true of troll patents, which tend to be asserted in the last few years of patent life.[193] Once those patents are understood to invoke Section 112(f), their literal scope will be limited to the technology the patentee actually designed and equivalents known at the time the patent issued.[194]

It is true that Section 112(f) equivalence is treated as literal infringement, raising the possibility that there could be an "equivalent to the equivalent" under the traditional "doctrine of equivalents."[195] But courts have read the doctrine of equivalents narrowly in the last fifteen years, to such an extent that the ordinary doctrine of equivalents has diminished to near the vanishing point.[196] While in part that results from judicial limits on the doctrine of equivalents that do not apply as readily to Section 112(f) equivalents,[197] some important limits, such as the rule against expanding the patent to cover the prior art, will apply to means-plus-function claims as well as to the normal doctrine of

192. *See* Cohen & Lemley, *supra* note 100, at 1762.

193. *See* Brian J. Love, *An Empirical Study of Patent Litigation Timing: Could a Patent Term Reduction Decimate Trolls without Harming Innovators?*, 161 U. PA. L. REV. 1309, 1336–40 (2013).

194. That won't solve the problem entirely, because some patentees will use the ability to file an unlimited number of continuation applications to delay issuance of a patent until years after invention. *See, e.g.*, Lemley & Moore, *supra* note 145, at 79–80. And for purposes of Section 112(f) it is the day the patent issues, not the day it is filed, that is relevant for determining the range of equivalents. *See* Lemley, *Changing Meaning*, *supra* note 191, at 103. An effort by the PTO to put some limits on the scope of continuations was challenged in court and ultimately withdrawn in 2009. *Tafas v. Kappos*, 586 F.3d 1369, 1371 (Fed. Cir. 2009) (en banc).

195. No, really, that's the rule. Ain't patent law grand?

196. *See, e.g.*, Allison & Lemley, *supra* note 60; Lee Petherbridge, *On the Decline of the Doctrine of Equivalents*, 31 CARDOZO L. REV. 1371 (2010); David L. Schwartz, *Explaining the Demise of the Doctrine of Equivalents*, 26 BERKELEY TECH. L.J. 1157 (2011). For debates over whether the demise of the doctrine of equivalents is good or bad, compare Meurer & Nard, *supra* note 19, with Doug Lichtman, *Substitutes for the Doctrine of Equivalents: A Response to Meurer and Nard*, 93 GEO. L.J. 2013 (2005). For a discussion specific to software and later-developed technology, see Cohen & Lemley, *supra* note 100, at 53–56.

197. For instance, the doctrines of prosecution history estoppel and dedication to the public domain are based on changes in the scope of the patent claim during prosecution. *See, e.g.*, *Festo Corp. v. Shoketsu Kinzoku Kogyo Kabushiki Co.*, 535 U.S. 722, 733–35 (2002); *Warner-Jenkinson Co. v. Hilton Davis Chem. Co.*, 520 U.S. 17, 30–34 (1997); *Johnson & Johnston Assocs. v. R.E. Serv. Co.*, 285 F.3d 1046, 1052 (Fed. Cir. 2002) (en banc) (per curiam). Because the technology disclosed in the specification doesn't change, those doctrines generally will not limit Section 112(f) structural equivalents.

equivalents.[198] In any event, the Federal Circuit is not inclined to read equivalents claims broadly because doing so undermines whatever notice function claims serve and makes it harder to resolve legal questions without a jury trial.[199] It has avoided applying other doctrines that give similar flexibility to the doctrine of equivalents, such as the reverse doctrine of equivalents[200] and the pioneer patents rule.[201] As a result,

198. *Wilson Sporting Goods Co. v. David Geoffrey & Assocs.*, 904 F.2d 677, 683–84 (Fed. Cir. 1990). Prosecution history estoppel too will still have some application to Section 112(f) equivalents because an applicant can estop themselves from claiming new ground on the basis of arguments to the PTO as well as claim changes. *See, e.g.*, *Aquatex Indus., Inc. v. Techniche Solutions*, 419 F.3d 1374, 1382 (Fed. Cir. 2005) ("The doctrine of prosecution history estoppel limits the doctrine of equivalents when an applicant . . . clearly and unmistakably surrenders subject matter by arguments made to an examiner." (quoting *Salazar v. Procter & Gamble Co.*, 414 F.3d 1342, 1344 (Fed. Cir. 2005)); *Pharmacia & Upjohn Co. v. Mylan Pharm., Inc.*, 170 F.3d 1373, 1376–77 (Fed. Cir. 1999).

199. *See, e.g.*, *Sage Prods., Inc. v. Devon Indus., Inc.*, 126 F.3d 1420, 1424–25 (Fed. Cir. 1997). The Federal Circuit has shown a strong preference for rules over standards in patent law in general. *See, e.g.*, Duffy, *supra* note 134, at 611; Holbrook, *supra* note 168, at 2; Lee, *supra* note 168, at 46; Craig Allan Nard, *Legal Forms and the Common Law of Patents*, 90 B.U. L. REV. 51, 84 (2010); Osborn, *supra* note 168, at 421–22; Thomas, *supra* note 168, at 792.

200. The reverse doctrine of equivalents constitutes an optional component of literal claim analysis, relieving the accused infringer of liability if the accused device, despite falling within the literal scope of the claims, is so far changed in principle that it performs a different function in a different way than the equivalent structure in the patent. The classic statement of reverse equivalents comes from *Boyden Power-Brake Co. v. Westinghouse*, 170 U.S. 537, 568 (1898). The doctrine is rarely applied, and the Federal Circuit in *Tate Access Floors, Inc. v. Interface Architectural Res., Inc.*, 279 F.3d 1357, 1368 (Fed. Cir. 2002), suggested that the doctrine had no continued meaning after the passage of the 1952 Patent Act. The court also (misleadingly) suggested the Federal Circuit had never applied the doctrine. *Cf. Scripps Clinic & Research Found. v. Genentech, Inc.*, 927 F.2d 1565, 1581 (Fed. Cir. 1991) (applying the reverse doctrine of equivalents). The Federal Circuit has since backed off from this crabbed and ahistorical reading. *See Amgen Inc. v. Hoechst Marion Roussel*, 314 F.3d 1313, 1351 (Fed. Cir. 2003).

201. The pioneer patent rule gave patents broader scope if they were pioneering inventions. *See, e.g.*, *Miller v. Eagle Mfg. Co.*, 151 U.S. 186, 207 (1894) ("If the invention is broad or primary in its character, the range of equivalents will be correspondingly broad, under the liberal construction which the courts give to such inventions."); *Perkin-Elmer Corp. v. Westinghouse Elec. Corp.*, 822 F.2d 1528, 1532 (Fed. Cir. 1987) ("A pioneer invention is entitled to a broad range of equivalents."). The Wright brothers, for example, won their patent infringement suit against Glenn Curtis in 1914 because they were pioneering inventors, and the court accordingly afforded them broad protection even against the somewhat different Curtis plane. *Wright Co. v. Herring-Curtis Co.*, 211 F. 654, 655 (2d Cir. 1914). The Court of Customs and Patent Appeals, the predecessor to the Federal Circuit, applied the pioneer patent doctrine, *see Autogiro Co. v. United States*, 384 F.2d 391, 400 (Ct. Cl. 1967), and the Supreme Court continues to talk about patent scope under the doctrine of equivalents as a function of how pioneering the patent is. *See Warner-Jenkinson*, 530 U.S. at 27 n.4. The pioneer

while the doctrine of equivalents means software patentees may sometimes get control over an entire function even under Section 112(f), those cases are likely to be quite rare. On balance, limiting functional software claims to the algorithm the patentee actually developed and equivalents thereof will go a long way towards narrowing the claimed scope of those patents, assuming they actually disclose such an algorithm. And if they don't, they are (and should be) invalid under *Aristocrat Technologies Australia PTY Ltd. v. International Game Technology*.[202]

2. DO INVENTORS DESERVE TO OWN FUNCTIONS?

A second class of objections to taking Section 112(f) seriously is in some sense the opposite of the first. This objection assumes that treating software patents as means-plus-function claims will in fact work, but it worries that doing so will unfairly disadvantage patentees. There are several species of this argument.

a. Hardware Doesn't Matter

Software, this argument goes, is all about the implementation of a function across machines. It shouldn't matter whether you want to run a spreadsheet on a PC, a Mac, an Android phone, or an old IBM mainframe. Each one might require a different computer implementation, but the genius of software is that those implementations are functionally equivalent; the machine is irrelevant. Thus, advocates of broad software patenting may argue that limiting them only to one particular algorithm or implementation in one particular machine unfairly restricts the scope

---

patent rule has not been invoked by the Federal Circuit in recent years, leading some to consider it moribund. *Compare Augustine Med., Inc. v. Gaymar Indus., Inc.*, 181 F.3d 1291, 1301 (Fed. Cir. 1999) (stating that "pioneering inventions deserve a broader range of equivalents"), *with Sun Studs, Inc. v. ATA Equip. Leasing, Inc.*, 872 F.2d 978, 987 (Fed. Cir. 1989) (holding that "the 'pioneer' is not a separate class of invention"), *overruled on other grounds*, *A.C. Aukerman Co. v. R.L. Chaides Constr. Co.*, 960 F.2d 1020 (Fed. Cir. 1992). The Federal Circuit did endorse the pioneering patent doctrine in an unpublished opinion in 2003. *See Molten Metal Equip. Innovations, Inc. v. Metallics Sys. Co.*, 56 F. App'x 475, 480 (Fed. Cir. 2003) (stating that pioneering invention claims "are entitled to a broad or liberal range of equivalents"). For discussion of the pioneer patent doctrine, see, for example, Meurer & Nard, *supra* note 19, at 2002–05 (endorsing broader use of the doctrine) and Thomas, *supra* note 19, at 37 ("Courts construe pioneer patent claims . . . to encompass a broader range of so-called 'equivalents' during an infringement determination."). *See also* Love, *supra* note 19 (arguing for its abolition).

202. 521 F.3d 1328, 1336–38 (Fed. Cir. 2008) (holding that an algorithm must be disclosed in order for a patent to be upheld).

of their patent, allowing other companies to avoid the patent while implementing an equivalent technology.[203]

There is something to this concern. In particular, it makes little sense to say that the implementation of the same algorithm in a different computer should be outside the scope of the patent.[204] As I suggested above, focusing on the hardware misses the point when the invention is one that is implemented in software. And patent claims are always cast at some level of abstraction away from the precise machine the patentee built so that they cover ideas rather than particular machines.[205] But when the patent seeks to cover not the implementation of a specific algorithm across different machines, but the implementation of different algorithms that happen to achieve the same end, that patent is too broad. It does not follow that because two algorithms solve the same problem that they are equivalent.[206] Thus, I part ways with those who argue that any invention in software is inherently an invention only at the level of the function it performs.[207] A moment's reflection on the history of software will reveal the flaw in that assumption. Google is a better search engine than its predecessors not because it performs a different function, but because it performs the same function in a different and better way. It is the way, not the function, that patent law is supposed to protect.

It is true that the different algorithm may compete with the patented one, preventing the patentee from excluding competition and raising prices. But so what? The vast majority of patents in all fields face some competition from other means of achieving the same end, and as a result

---

203. *See, e.g.*, Robert R. Sachs, *Comments in Response to the Patent and Trademark Office's Proposed Examination Guidelines for Computer-Implemented Inventions*, 2 MICH. TELECOMM. & TECH. L. REV. 103, 107 (1995–96); Note, *supra* note 61, at 1465–66 ("[I]t would make no sense for software patentees to specify secondary characteristics like a programming language, operating system, or platform in their patents. These have nothing to do with the invention.").

204. Note, *supra* note 61, at 1471 (arguing that "software patents are broad without being overclaimed"). For this reason, I have argued elsewhere that the "machine or transformation" test for patentable subject matter, which would limit software patents to those "tied to a particular machine," doesn't make sense. Lemley et al., *Life after* Bilski, *supra* note 132, at 1346–47.

205. *See* Dan L. Burk & Mark A. Lemley, *Quantum Patent Mechanics*, 9 LEWIS & CLARK L. REV. 29, 32–40 (2005) (noting the "levels of abstraction" problem in interpreting patent claims) [hereinafter Burk & Lemley, *Quantum*]; Tun-Jen Chiang, *The Levels of Abstraction Problem in Patent Law*, 105 NW. U. L. REV. 1097 (2011).

206. For a technical discussion in the field of software, see Andrew Chin, *On Abstraction and Equivalence in Software Patent Doctrine: A Response to Bessen, Meurer and Klemens*, 16 J. INTELL. PROP. L. 197 (2009).

207. Note, *supra* note 61, at 1474.

most patents don't confer market power.[208] If I invent a particular blade shape for a lawn mower, patent law gives me the right to prevent competitors from making a blade in that shape.[209] It doesn't give me a right to control lawn mowers generally; anyone who makes a differently shaped blade can sell it without infringing even if it performs the same function and does it just as well as the patented invention. Similarly, if I develop a cholesterol-reducing drug, I don't get to claim "atoms configured in a way that reduces human cholesterol." My patent is limited to the drug I actually make and others like it. Even if I am the first to develop a cholesterol-lowering drug, the fact that I can't claim the function itself leaves open the possibility that others will later develop different drugs that achieve the same end.[210]

That doesn't mean that the inventor's contribution should be limited to the precise code she wrote; the invention may well make a contribution at a higher level of abstraction.[211] And if it does, the patent can properly capture a group of related implementations of that same idea. But if "the idea" is "solve this problem," we should be very

---

208. *See, e.g.*, *Illinois Tool Works Inc. v. Independent Ink, Inc.*, 547 U.S. 28, 31 (2006) (rejecting prior presumption that patents necessarily confer market power). For criticism of the equation of patent and market power, see, for example, HERBERT HOVENKAMP ET AL., 1 IP AND ANTITRUST: AN ANALYSIS OF ANTITRUST PRINCIPLES APPLIED TO INTELLECTUAL PROPERTY LAW § 4.2 (2d ed. 2010). *Cf.* Louis Kaplow, *Why (Ever) Define Markets?*, 124 HARV. L. REV. 437, 500–01 (2010); Lemley & McKenna, *supra* note 25, at 2091 (pointing out that a surprising number of IP rights do in fact confer power over price).

209. This leaves open the question of the level of generality at which the technology is protected. A patent claim that was strictly limited to exactly the device the patentee built would be too easy to evade. So patentees are entitled to prevent others from implementing the concept of the invention even if the details differ. That is why patent claims cover a genus of implementations, not just a particular species. *See, e.g.*, Burk & Lemley, *Fence Posts*, *supra* note 15; Lefstin, *supra* note 22, at 1168–69. But the level of generality is never "the market"—except, that is, in software. *Cf.* Abramowicz & Duffy, *supra* note 25, at 340 (arguing for legal protection for market information).

210. As Justice Frankfurter wrote in 1948, concurring in the rejection of functional claims to a collection of bacteria,

> The consequences of such a conclusion call for its rejection. Its acceptance would require, for instance in the field of alloys, that if one discovered a particular mixture of metals, which when alloyed had some particular desirable properties, he could patent not merely this particular mixture but the idea of alloying metals for this purpose . . . In patenting an alloy, I assume that both the qualities of the product and its specific composition would need to be specified.

Funk Bros. Seed Co. v. Kalo Inoculant Co., 333 U.S. 127, 134 (1948) (Frankfurter, J., concurring).

211. On levels of abstraction in patent law, see Burk & Lemley, *Quantum*, *supra* note 205, at 32–40.

cautious about giving a patent at that high a level of abstraction. Doing so may give the patentee control over a market not because their product is superior to others in that market, but by definition—the market is the thing the patent itself claims.[212] And in software patents, all too often that is precisely what we have been patenting.

Patents, then, are not designed to control markets, though sometimes they do. Rather, they are designed to encourage the development of new inventions and differentiated products within that market by discouraging copying of the patentee's technology.[213] Those new inventions will often be imperfect substitutes, so patents will often confer some power over price.[214] But it has never been the purpose of patent law to give the patentee control over a market as opposed to a technology. Indeed, the economic evidence is pretty good that competition is itself a spur to new innovation.[215] The existence of a

212. *See* Collins, *supra* note 90, at 17–23 (explaining in detail the harms associated with claiming markets).

213. *See generally* Yoo, *Differentiation*, *supra* note 25.

214. Lemley & McKenna, *supra* note 25, at 2091.

215. *See, e.g.*, Juan A. Correa & Carmine Ornaghi, Competition & Innovation: New Evidence from US Patent and Productivity Data (Oct. 21, 2011) (working paper), *available at* http://ssrn.com/abstract=1947357 (finding that, across industries, innovation is faster in more competitive markets). For a sense of the literature on this long-running economic debate, see Kenneth J. Arrow, *Economic Welfare and the Allocation of Resources for Invention*, *in* NAT'L BUREAU ECON. RESEARCH, THE RATE AND DIRECTION OF INVENTIVE ACTIVITY: ECONOMIC AND SOCIAL FACTORS 609, 620 (1962) (concluding that "preinvention monopoly power acts as a strong disincentive to further innovation"). *See also* MORTON I. KAMIEN & NANCY L. SCHWARTZ, MARKET STRUCTURE AND INNOVATION 16 (1982) (discussing various theories of the effects of economic structures on the rate and form of innovation); F.M. SHERER & DAVID ROSS, INDUSTRIAL MARKET STRUCTURE AND ECONOMIC PERFORMANCE 660 (3d ed. 1990) (criticizing Schumpeter's "less cautious" followers for advocating monopoly to promote innovation). In the specific context of IP, the canonical argument from both theory and empirical evidence is Robert P. Merges & Richard R. Nelson, *On the Complex Economics of Patent Scope*, 90 COLUM. L. REV. 839 (1990). *See also* Kenneth W. Dam, *The Economic Underpinnings of Patent Law*, 23 J. LEGAL STUD. 247, 252 (1994) (noting that in the computer industry, for example, companies coordinate improvements by broad cross-licensing because of "the pace of research and development and the market interdependencies between inventions"). For discussions of particular industries in which competition appears to spur innovation, see, for example, Arti Kaur Rai, *Evolving Scientific Norms and Intellectual Property Rights: A Reply to Kieff*, 95 NW. U. L. REV. 707, 709–10 (2001) (biotechnology); Mark A. Lemley & Lawrence Lessig, *The End of End-to-End: Preserving the Architecture of the Internet in the Broadband Era*, 48 UCLA L. REV. 925, 960–62 (2001) (the internet); Howard A. Shelanski, *Competition and Deployment of New Technology in U.S. Telecommunications*, 2000 U. CHI. LEGAL F. 85, 85 (telecommunications).

patent on one technology might spur design-arounds that lead to new inventions that compete (imperfectly) with the patented one.[216]

b. Point of Novelty

A different argument is that we shouldn't care if the standard, non-inventive elements of the invention known in the prior art are described in functional terms. If the novelty in an invention lies in the encryption algorithm used, it shouldn't matter that the processor on which the algorithm runs, or the database in which the keys are stored, are described in functional terms because the value of the invention is the same regardless of which processor or database the user employs.

I agree with this concern; I have complained for years that we focus too much attention on the often-trivial language of the patent claims and not enough attention on the actual novel piece of the patentee's invention.[217] But even in the current regime, with its hyper-technical focus on the language of the claims, Section 112(f) can accommodate this concern. Standard computing elements are precisely the sort of things that ought to be written in means-plus-function language. The fact that it doesn't matter what database an inventor uses in his encryption program—that for his purposes they are all equivalent—will mean that the claim elements not located at the point of novelty will be entitled to broad construction.[218] It is precisely at the point of novelty that the

---

216. On the economic benefits of design-arounds, see, for example, *Warner-Jenkinson Co. v. Hilton Davis Chem. Co.*, 520 U.S. 17, 36 (1997) (contrasting "the intentional copyist making minor changes to lower the risk of legal action" with "the incremental innovator designing around the claims, yet seeking to capture as much as is permissible of the patented advance."); *see also Slimfold Mfg. Co. v. Kinkead Indus., Inc.*, 932 F.2d 1453, 1457 (Fed. Cir. 1991) ("Designing around patents is, in fact, one of the ways in which the patent system works to the advantage of the public in promoting progress in the useful arts, its constitutional purpose."); *State Indus., Inc. v. A.O. Smith Corp.*, 751 F.2d 1226, 1236 (Fed. Cir. 1985) ("One of the benefits of a patent system is its so-called 'negative incentive' to 'design around' a competitor's products, even when they are patented, thus bringing a steady flow of innovations to the marketplace."); Nard, *supra* note 179, at 40–41 ("The practice of designing-around extant patents creates viable substitutes and advances, resulting in competition among patented technologies. The public clearly benefits from such activity.")

217. *See* Burk & Lemley, *Fence Posts*, *supra* note 15; Lemley, *Novelty*, *supra* note 186.

218. *See, e.g.*, *IMS Tech., Inc. v. Haas Automation, Inc.*, 206 F.3d 1422, 1436 (Fed. Cir. 2000) (a claim element of little importance to the invention—not at the point of novelty—is entitled to a broader range of equivalents). In a once-common form of claim called a "Jepson claim," patentees would identify in the preamble the existing technology and then identify their improvement in the body of the claim. *See Ex parte Jepson*, 1917 Dec. Comm'r Pat. 62, 67–68. This had the benefit of highlighting what was actually new about the patentee's invention. In a *Jepson* claim, the distinction could be quite clear:

patentee should be forbidden from substituting broad functional language for an actual implementation of the invention. I should be able to include "an analog-to-digital converter" in my claim if ADCs are well known in the art and not the focus of my invention, but if I am the first person to have invented a way of converting data from analog to digital format, I shouldn't be allowed simply to claim "an analog-to-digital converter" without any limitation as to how the invention works. The same is true in software.[219]

c. Software Inherently Functions

A broader objection is that software ought to be different. The most common variant of this claim is that software is inherently functional. Thus, Kevin Collins argues that inventions in the software arts are pure functionality: that they can only be defined by their functional properties, as their physical, structural properties have no relevance to the definition of what an inventor has invented.[220] Collins may mean only that hardware implementations don't matter; there, I agree. But I think he intends something deeper—that an algorithm, at least itself defined sufficiently broadly, is a series of steps, and those steps are themselves best understood in functional terms.[221] Kip Werking goes further, arguing that "the ultimate problem for the algorithm requirement [is that] algorithms *are composed of* functions."[222] At a low enough level, that is clearly untrue. Software "functions" because it causes a series of gates in a computer chip to open and close in a particular sequence. At that level,

functional description is permissible in the preamble, because that's not what the patentee invented, but would not be permissible in the identified improvement. Unfortunately, *Jepson* claiming is on the decline. While *Jepson* claims represented 15% of all claims thirty years ago, they are less than 1% today. Aaron R. Feigelson, *Endangered Species: The Jepson Claim*, 12:01 TUESDAY, http://www.1201tuesday.com/1201_tuesday/2009/06/jepson.html (last updated June 4, 2009, 1:56 PM).

219. Thus, the Federal Circuit has held that a patentee need not disclose a particular algorithm "if the selection of the algorithm or group of algorithms needed to perform the function in question would be readily apparent to a person of skill in the art." *Aristocrat Techs. Austl. Pty Ltd v. Multimedia Games*, *Inc.*, 266 F. App'x. 942, 947 (Fed. Cir. 2008).

220. Collins, *supra* note 90, at 4.

221. Kip Werking makes this argument. Kip Werking, *The Illogic of the Algorithm Requirement for Software Patent Claims*, IPWATCHDOG (Oct. 12, 2012, 7:20 AM) http://www.ipwatchdog.com/2012/10/12/the-illogic-of-the-algorithm-requirement-for-software-patent-claims/id=28635/.

222. *Id. Cf.* Andrew Chin, Alappat *Redux: Support for Functional Language in Software Patent Claims* (UNC Legal Studies Research Paper No. 2272016, 2013), *available at* http://papers.ssrn.com/sol3/papers.cfm?abstract_id=2272016 (arguing that "algorithm" is "too slippery" a concept to serve as the basis for means-plus-function patenting and proposing a philosophical inquiry into single causation instead).

software operates a machine by changing its physical orientation. That's not functional; that's a device.[223] True, we don't want to limit the patentee to the machine-level implementation. But we don't have to jump immediately to function as we go up the level of abstraction. Object code is a representation of that programmed device; at a slightly higher level of abstraction, so is source code. But they are representations of an ultimately material thing—the programmed computer. The same is true as we go further up the level of abstraction. A dynamic linked list, for example, is a well-understood class of software objects. We could say it is "functional" because the class is defined in part by whether different objects perform the same function. But I don't think that's a meaningful way to think about it. It is true at some philosophical level but need not paralyze us. Indeed, it's true of any patent claim that covers a genus of things (which is to say every patent claim, as Jeff Lefstin reminds us).[224] A patent claim to a "chair comprising a seat, a back, and a plurality of legs" includes concepts—like the "seat"—that unite otherwise-disparate things by the function they perform. A jackhammer functions too, but we have no trouble distinguishing the function it performs from the way in which it performs that function. The same can be said of software. Saying that a claimed computer program must use functional language because it speaks of a dynamic linked list ignores the fact that we understand the term "dynamic linked list" to refer to a specific class of software objects, just as we understand the term "seat" (or "resistor" or "analog-to-digital converter" or "timing circuit") to refer to a specific class of things.

To the extent Collins and Werking mean that one cannot conceptually distinguish one software approach from another, I disagree. There are clearly different ways of solving a problem in software that map to different, well-understood software objects and subroutines, and they may have different advantages or disadvantages in terms of ease of construction, stability, speed, and output. Function is simply not the same as implementation. And distinguishing between different programs that perform the same function in a different way is precisely what patent law is supposed to do. If you implement sorting using a quicksort algorithm, you are entitled to claim the use of a quicksort algorithm but not the idea of sorting in any way whatever.

---

223. Thus, the Federal Circuit concluded in *In re Alappat*, 33 F.3d 1526, 1545 (Fed. Cir. 1994) (en banc) that a computer programmed with new software becomes for all intents and purposes a new machine because the hardware itself is modified by the program.

224. Lefstin, *supra* note 22.

None of that enables us to avoid the hard work of choosing a level of abstraction. If we aren't to limit the patent to the exact code the patentee used (and we shouldn't), we will need to find an intermediate level of abstraction in which the program is decomposed into algorithmic steps that are themselves understood to have particular meanings. That won't always be easy; courts and lawyers will doubtless disagree over what the algorithm in the specification is, just as they disagree about what structure corresponds to any other functional patent claim element.[225] But that isn't a reason to ignore the language of the statute. And, as I have suggested elsewhere, it is a general problem for patent scope, not a particular problem with software patents.[226]

d. The Function Is the Invention

Finally, at least some patentees will claim that the programming didn't matter and the discovery of a new function was itself the invention. But while that may be true in some cases (though surely it is not true of most software inventions), there is good reason to think that in software in particular, it is competition and not market dominance that spurs innovation.[227] And Part III offers us compelling reasons to believe that giving such broad functional patents in software causes major problems for the patent system.[228] So even if we thought that, as a matter of logic, software patents should be different than other kinds of patents, on balance it seems a mistake to permit broad functional claiming of software. Software inventors could still claim genuses; they should not be limited to the precise code they wrote. But the patent must be limited to the actual technology the patentee developed, defined at an appropriate level of abstraction, not to the problem it addressed, however solved. If the objection is that applying Section 112(f) limits patentees to the technology they actually designed or similar ones, then the answer is: too bad. Patent law is, after all, designed to benefit society, not just the patentee.[229]

---

225. For some suggestions of ways to draw these lines, see Edlin, *supra* note 88.

226. *See* Burk & Lemley, *Quantum*, *supra* note 205, at 31 (arguing that claims must be understood at a particular level of abstraction, and the law currently doesn't recognize the choices it is making); Tun-Jen Chiang, *supra* note 205, at 1101–02.

227. *See* BURK & LEMLEY, PATENT CRISIS, *supra* note 124, at ch.5 (arguing that free competition may best promote Internet innovation and that narrow patents will do so in cumulative innovation industries like software).

228. *See supra*, Part III.

229. *See, e.g.*, *Sears, Roebuck & Co. v. Stiffel Co.*, 376 U.S. 225, 229–30 (1964) ("Patents are not given as favors . . . but are meant to encourage invention by rewarding the inventor . . . ."); Lemley, *Free Riding*, *supra* note 147, at 1072–73; Ted Sichelman,

3. LIMITING SOFTWARE PATENTS IN ORDER TO SAVE THEM

In fact, it may ultimately be the case that software patentees also stand to benefit from the application of Section 112(f). In the last four years, the courts have begun enforcing strict limits on patentable subject matter in software cases. In *Bilski v. Kappos*,[230] the Supreme Court held that a business method patent was unpatentable as an abstract idea because it was not sufficiently tied to a particular real-world implementation.[231] In the wake of that decision, most (though not all) Federal Circuit decisions to consider the patentability of software have held that software patents that merely implemented process steps in a general-purpose computer were unpatentable because the process steps alone were too abstract, and the presence of a general-purpose computer was insufficient to limit the claim to a particular real-world implementation.[232] While the law of patentable subject matter is still unsettled, the current trend is one that would invalidate a wide swath of software patent claims, particularly functional claims of the type I consider here—not because they are too broad, or indefinite, but because they are not the sort of thing that is patentable at all.

Treating these functional software patent claims as means-plus-function claims may end up saving them from invalidation under Section 101. If the patent is interpreted as a means-plus-function claim, it will be limited to the particular software implementation the patentee actually built or described. Such a narrow, specific claim should not be an unpatentable "abstract idea."[233] And focusing on the actual

---

Purging Patent Law of "Private Law" Remedies, 92 TEX. L. REV. (forthcoming 2013), *available at* http://papers.ssrn.com/sol3/papers.cfm?abstract_id=1932834.

230. 130 S. Ct. 3218 (2010).

231. *Id.* at 3230–31.

232. *See, e.g.*, *Accenture v. Guidewire, Inc.*, No. 11-1486 (Fed. Cir. Sept. 5, 2013); *CLS Bank v. Alice Corp. Pty.*, No. 11-1301 (Fed. Cir. May 10, 2013) (en banc) (per curiam); *Fort Props., Inc. v. Am. Master Lease LLC*, 671 F.3d 1317, 1323–24 (Fed. Cir. 2012) (rejecting as unpatentable claims that listed functional steps implemented in a general-purpose computer); *Dealertrack, Inc. v. Huber*, 674 F.3d 1315, 1332–33 (Fed. Cir. 2012); *CyberSource Corp. v. Retail Decisions, Inc.*, 654 F.3d 1366, 1375 (Fed. Cir. 2011); *cf. Ultramercial, LLC v. Hulu, LLC*, 722 F.3d 1335, 1350 (Fed. Cir. 2013) (holding that a method of implementing process steps using the Internet was patentable because it was likely to involve "complex computer programming"). *But see Research Corp. Techs. v. Microsoft Corp.*, 627 F.3d 859, 867–69 (Fed. Cir. 2010) (holding claim to method of controlling computer display patentable subject matter). For discussion, see Pamela Samuelson & Jason Schultz, *"Clues" for Determining whether Business and Service Innovations Are Unpatentable Abstract Ideas*, 15 LEWIS & CLARK L. REV. 109 (2011).

233. My co-authors and I have argued that elsewhere. *See* Lemley et al., *Life after* Bilski, *supra* note 132.

software invention will move the patentable subject matter law away from unhelpful tests like the "machine or transformation" test that focus on the hardware, not the software, in the patent claims.[234] Those narrowed patents are also less likely to be invalid on the basis of prior art, since it is far more likely that someone has described the same function before than that they have produced the same algorithm before. And developing the algorithm itself may well be nonobvious in many cases.[235]

Restricting functional claiming, then, may have the unexpected effect of saving many software patents from invalidation by narrowing them.[236] Opponents of software patents may think that a problem; they are hoping that the new patentable subject matter cases will invalidate all software patents. But I think it's a good thing. There is nothing wrong with the idea of patenting true inventions in software; the problem lies in the overclaiming we have permitted in the current system. If we can get rid of that overclaiming, we can limit software patents to what the patentees actually invented, encouraging genuine innovation without promoting patent holdup.

An algorithm requirement is not a panacea.[237] Eliminating pure functional claiming will not automatically teach us how broad a software patent is or whether a defendant infringes. Patentees will be entitled not just to the precise algorithm they used, but to sufficiently similar algorithms, and courts will have to assess that similarity, just as they do in every other area of patent practice.[238] But focusing on what the patentee and the defendant actually did, rather than the problem they solved, will cabin the range of debate and limit over-claiming of software. They will allow patent law to do what it is supposed to do—determine the right level of abstraction at which the patentee can claim the invention.

---

234. For an argument that this hardware focus is misguided in Section 101 analysis, see Bernard Chao, *Finding the Point of Novelty in Software Patents*, 28 BERKELEY TECH. L.J. (forthcoming 2013), *available at* http://papers.ssrn.com/sol3/papers.cfm?abstract_id=2174527.

235. Fromer, *Layers*, *supra* note 141, at 79–82.

236. If the claims never disclose any algorithm or other implementation, they will be invalid for indefiniteness under *Aristocrat* and its progeny. *See supra* note 202 and accompanying text. But that is as it should be; a patent that doesn't disclose any implementation of the idea, but merely the functional goal, doesn't have at its heart a real invention.

237. Collins, *supra* note 90, at 62–72.

238. *See* Burk & Lemley, *Quantum*, *supra* note 205; Chiang, *supra* note 205.

CONCLUSION

It is time to end functional claiming (again). Allowing inventors to assert ownership over the problem they solved, rather than merely the way they solved it, is inconsistent with history, with the patent statute, and with good patent policy. It is responsible in large part for the untenable situation software patents have left us in. And while software patent owners may object that they need functional claiming to get effective protection, that objection is unpersuasive, both because of the harm functional claiming causes and because functional patent claims are likely invalid under current law.

A patent should not guarantee insulation from competition. To the contrary, properly understood, patents spur competition by preventing direct imitation while leaving open avenues for alternative development. We have forgotten that lesson in software, to our great cost. Returning to a world in which inventors own their idea, but not the ideas of others, will go a long way towards ensuring that patents encourage rather than retard software innovation.