No. 08-964

IN THE

SUPREME COURT OF THE UNITED STATES

BERNARD L. BILSKI AND RAND A. WARSAW,

Petitioners,

v.

DAVID J. KAPPOS, UNDER SECRETARY OF COMMERCE FOR INTELLECTUAL PROPERTY AND DIRECTOR, PATENT AND TRADEMARK OFFICE,

Respondent.

On Writ of Certiorari to the United States Court Of Appeals for the Federal Circuit

BRIEF OF AMICI CURIAE

FOUNDATION FOR A FREE INFORMATION INFRASTRUCTURE, IP JUSTICE, AND FOUR GLOBAL SOFTWARE PROFESSIONALS AND BUSINESS LEADERS IN SUPPORT OF RESPONDENT

Allonn E. Levy
Hopkins & Carley
408.286.9800 (main)
70 South First Street
San Jose, CA 95113
COUNSEL OF RECORD

TABLE OF CONTENTS

Page

TABLE OF AUTHORITIES
(Continued)

Page

TABLE OF AUTHORITIES

Page

Cases

Statutes

Other Authorities

TABLE OF AUTHORITIES
(Continued)

Page

Rules

I.

QUESTIONS PRESENTED

1. Whether the Federal Circuit erred by holding that a "process" must be tied to a particular machine or apparatus, or transform a particular article into a different state or thing (Machine-or-Transformation test), to be eligible for patenting under 35 U. S. C. §101, despite this Court's precedent declining to limit the broad statutory grant of patent eligibility for "any" new and useful process beyond excluding patents for "laws of nature, physical phenomena, and abstract ideas".

2. Whether the Federal Circuit's Machine-or-Transformation test for patent eligibility, which effectively forecloses meaningful patent protection to many business methods, contradicts the clear Congressional intent that patents protect "method[s] of doing or conducting business." 35 U. S. C. § 273.

II.

STATEMENT OF INTEREST OF AMICI CURIAE[1]

A. About the Foundation for a Free Information Infrastructure

The Foundation for a Free Information Infrastructure (FFII) e. V. is a charitable association registered in Munich, Germany which is dedicated to the spread of data processing literacy. It funds the development of public information works based on copyright, free competition and open standards.

The FFII attained broad international recognition for its phrontistery role in the European debate on a software patent directive (2002-2005) and software-related patent reform. It is a registered observer at the World Intellectual Property Organization.

The association and its members aim to reduce friction costs and risks for software authors and to prevent dilution of property rights under the Berne Convention by territorial patent grants. To this end FFII is guided by

---

[1] Pursuant to Sup. Ct. R. 37.6, amicus notes that no counsel for a party authored this brief in whole or in part, and no counsel or party made a monetary contribution intended to fund the preparation or submission of this brief. No person other than amicus curiae, its members, or its counsel made a monetary contribution to its preparation or submission. Petitioners and Respondents have consented to the filing of this brief through blanket consent letters filed with the Clerk's Office.

Hayek's dictum that "it is necessary [. . .] not to apply a ready-made formula but to go back to the rationale of the market system and to decide for each class what the precise rights are to be which the government ought to protect. This is a task at least as much for economists as for lawyers."[2]

For digital markets, cheap, fast and narrow rights are sought. The interests of the association are improvements of the substantive patent rules and the examination process. A global challenge is to keep patent office bureaucracies manageable, and make them adapt to an acceleration of markets.

The FFII members who contributed to this Amicus Curiae Brief are entrepreneurs and programmers. They devoted their time to this project because they are directly affected by the economic impacts of patents.

B. About The Four Global Software Professionals and Business Leaders

Dr. rer. nat. Peter Gerwinski is an entrepreneur, software developer and physicist from Essen, Germany. He is founder and Managing Director of G-N-U GmbH and a contributor to the GNU Pascal programming language.

Laura Creighton is a Canadian-born entrepreneur and software developer from Gothenburg, Sweden. She is founder and Managing Director of Open End AB, which has developed the Eutaxia task organizing software, and engages in computer science research in the field of just-in-

---

[2] Hayek, F. A. v., Individualism and Economic Order, Chicago, University of Chicago Press, 1949, 114

time compiler generation for dynamic programming languages.

Mag. iur. Georg Jakob is a lawyer, business consultant and former University researcher and teacher from Salzburg, Austria, now living and working in Munich, Germany.

Dipl. Kfm. & M.A. André Rebentisch is a business intelligence and technology research specialist, currently residing in Wilhelmshaven, Germany.

These four business leaders are long-time FFII members and founders of the FFII's Global Patent Policy Research Group. Each has personally been affected by the beneficial and harmful effects of patents. Each joins this brief to voice concern about the dangers associated with unfettered grants of software and business method patents.

C. About IP Justice

IP Justice is an international civil liberties organization that promotes balanced intellectual property rights and protects freedom of expression. IP Justice is based in Silicon Valley, California with regional membership throughout the world. IP Justice is concerned about the potential impact the case pending before this Court may have on global policies, international treaties and trade agreements that address intellectual property rights. IP Justice has an interest in ensuring that traditional global limits are maintained on the reach of patent law to ensure that the proper balance is struck between the protection of past innovations without unduly hampering future innovations. It is precisely this delicate balance that is placed risk by proponents urging this Court to expand patent rights beyond the traditional time-

honored rules set forth in the U.S. constitution and the long-standing precedent of this court.

IP Justice participates in a number of international law and policy arenas including the World Intellectual Property Organization (WIPO), the Internet Corporation for Assigned Names and Numbers (ICANN), and the United Nations Internet Governance Forum (IGF) among others. IP Justice has twice been called to testify at hearings before the US Copyright Office on the anti-circumvention provisions in the US Digital Millennium Copyright Act (DMCA). IP Justice has an international board of directors and is 501(c)(3) nonprofit public benefit organization that was founded in 2002 by its current Executive Director Robin Gross.

III.

SUMMARY OF ARGUMENT

The Court of Appeals for the Federal Circuit in *In Re Bilski* wisely followed the Machine-or-Transformation test to avoid running afoul of the constitutional and precedential safeguards in place to protect the free exchange of abstract ideas in the marketplace. These safeguards are particularly important when analyzing non-physical innovations such as software and business method innovations.

The properties of software and business methods are fundamentally different from those of machines and transformations. Economic and anecdotal evidence provides strong justification for the conclusion that certain patent claims on software algorithms, business methods, and other abstract matter – untethered to any machine or

transformation – would stifle rather than promote the progress of the sciences and useful arts.

The Machine-or-Transformation test therefore provides a means to an end: A clear limit to patentable subject-matter in accordance with case law, which consistently rejects the patentability of *abstract ideas*. To avoid circumvention of the rule prohibiting patents on abstract ideas, it is necessary to apply patentability standards *to the claimed object* rather than to the patent claim *as a whole*:

1. An object that consists only of laws of nature, physical phenomena, and abstract ideas cannot be claimed in a patent, independent of the form in which it is claimed.

2. An object can only be claimed in a patent if it constitutes a new concrete realization of a machine or transformation.

Amici conclude this from their experience with the European *technicity* test which is discussed in greater detail below.

Plainly, with the potential reward of an unfettered monopoly, there exists great pecuniary incentive for special interests to attempt to place their thumb on the scale of justice in order to obtain that which has historically been prohibited. This court has long provided the guiding hand protecting society from the ills that would result from expanding the exclusionary power of patents beyond the boundaries set forth by the founding fathers and subsequent legislators. Amici now urge this court to continue that tradition and affirm the Machine-or-

transformation test and to include certain additional safeguards that are set forth below.

IV.

ARGUMENT

In a globalized world, business activities crossing the oceans are commonplace, even for small enterprises. This is especially true in the software market where software can be transferred between continents at no noticeable cost via the Internet. Consequently, even a one-person software company serves customers around the globe and is affected by territorial patents. But there is an even more clear connection between U.S. Patent law and the laws of foreign countries. Countries the world over perpetually work to harmonize patent laws world-wide. The European Patent Office (EPO) is currently revising its strategy with respect to patents on software and business methods. The outcome of the *Bilski* case will be carefully followed and will necessarily affect the EPO's decisions on these important issues.

The *Bilski* case reaches the question of patentability of business methods and by extension other non-tangible subject matter such as software that is not connected to any machine or transformative use. This affects virtually everyone doing business in the United States or with the United States. If innovators are permitted to monopolize business practices, algorithms, or abstract ideas, untethered to any physical mechanism the damaging effects of these unconstrained monopolies will be far-reaching. There is good reason that this High Court and

European Parlament have both, thus far, rejected such a rule.

A. The Lower Court's test Properly Excludes Abstract Ideas From the field of Patentable Material.

Nearly 150 years ago this Court stated the truism: "An idea of itself is not patentable." *Rubber-tip Pencil v. Howard,* 87 U.S. (20 Wall.) 498, 507 (1874). For decades this court has held that subject matter that would effectively exclude market participants from employing laws of nature, abstract ideas, or mathematical algorithms cannot be protected by patent[3]. Conceptually these rules are clear. However, like any rules, as the marketplace of innovation changes, the same rules must be applied to different technological advances.

1. Key Distinctions Between Hardware And Software Patents.

Over the last decades, computer software has become a pervasive and ubiquitous tool in the global economic toolbox. Although the development of mechanical parts has been optimized over thousands of years, software is gaining ever increasing importance. The reason for this increase in innovation lies at least in part in the distinctions between hardware and software. Distinctions that are pivotal to an understanding of why software, on its own, has traditionally not be afforded patent protection.

---

[3] *See e.g. Gottschalk v. Benson,* 409 U.S. 63 (1972), *Microsoft v. AT&T* 550 U.S. 437, 449 (2007).

Ease of Creation: One reason why software is being developed and propagated at a dramatic pace is that software is much easier, and therefore less costly, to create than hardware. Plainly, software is nothing more than a set of written instructions. By contrast, when a mechanical device is built, one has to contend with the laws of physics. No mechanical component is 100 % reliable. All lengths, diameters, etc. come with tolerances. Software also consists of components, but those components come in the form of "algorithms." Unlike physical components, algorithms are simply idealized mathematical expressions. These mathematical expressions have no tolerances or abrasion and are ubiquitous in the marketplace of ideas.

Ease of Scalability: Because of the differences in components, a mechanical device with 100 gears needs a physical redesign when a 101st gear needs to be added. This necessarily involves an added cost. By contrast, a piece of software can be extended to 10,000 "virtual gears" with no noticeable effort. It requires only the direction of a gifted programmer to provide the necessary expressions.

Ease of Portability: Because software is a set of instructions or expressions, those expressions can be easily manipulated and applied to accomplish nearly limitless tasks. These "expressive parts" are therefore very flexible and can be used in very different ways. Many of them can be used in fundamentally different types of software. For example, the same compression algorithm can be used in a drawing program, a cryptography system, a web server, a device driver, and much more.

In software development, it is commonplace to combine large existing parts from diverse sources —which in turn themselves consist of smaller and smaller parts —

to form an even larger system. For instance, you can create a minimalistic, but useable content management system by just putting together a file server and a web server. With hardware, it is much more difficult to combine the features of two completely different existing systems. For example, one cannot simply weld together a ship and an aircraft and expect the result to swim and to fly – while with software one *can* effectively create the analogous feat[4].

Ease of Duplication: The manufacturing of software is fundamentally different from that of mechanical devices. One important difference is that software can be copied without material degradation at no significant expense to the inventor[5]. In contrast, mechanical devices must be built one-by-one, often using materials and equipment which can pose at least some expense to the innovator.

a. Software And Business Methods Bear Little Risk of Unauthorized Reverse Engineering.

Some have argued that innovations in software require patent protection to guard against unauthorized

---

[4] See also http://www.gnu.org/philosophy/software-patents.html.

[5] As discussed below, while ease of copying is often a concern for those seeking to protect their software innovations from unauthorized duplication, there exist robust legal protections against such unauthorized duplication. The additional layer of protection that would be offered by patent protection (if it were made available) would add no additional disincentive to the unscrupulous infringer.

reverse engineering[6]. As demonstrated below, that is not the case.

Once an innovative automobile has been built, a competitor can save R&D expenses by purchasing one automobile and disassembling it to see how it works. However, the same is not possible with software. As an example, consider a software which calculates the surface A of a sphere with known radius r using the well-known formula A = 4 pi r^2.

To create this software, the author writes the so-called source code in a programming language. In the programming language "C" (ISO/IEC 9899:1999) this software can be written as follows:

```
 /* sphere.c - Calculate the Surface of a
Sphere */

 #include <stdio.h>

 #define pi 3.14159265

 int main (void)

 {

   float radius, surface;

   scanf ("%f", &radius);  /* read the radius */

   surface = 4.0 * pi * radius * radius;
```

[6] See for example the Brief of Amicus Curiae International Business Machines Corporation In Support of Neither Party filed in this action at fn. 22, p.23.

```
  printf ("%f\n", surface);  /* show result */

  return 0;  /* report success */

}
```

This source code cannot directly be executed by a computer. To make it executable, it is necessary to transform - to compile - it into binary code, which is a sequence of numbers which is difficult for humans to read.

A software developer who does not want to disclose how the software works or wants to retain the privilege to do any modifications on it, only distributes the binary code and keeps the source code as a trade secret.

There is in fact a process called "disassembling the software" which means to extract information out of the binary code. However, it is not feasible to retrieve a useful source code through disassembly. The following is the "C" equivalent of what can be extracted from the binary code of the "sphere" software above.

```
int main (void)

{

  float x1, x2;

  f1 ("%f", &x1);

  x2 = 12.5663706 * x1 * x1;

  f2 ("%f\n", x2);

  return 0;
```

```
}
```

In this "reconstructed source code" some important pieces of information are irretrievably lost,

- all comments (/* ... */ in "C"),

- the names "radius", "surface" of the variables,

- the names "scanf", "printf" of the functions, an

- the origin "4 pi" of the number "12.5663706".

Even in this most simple example, it would require extreme expertise and substantial guesswork or trial-and-error to recover the meaning of "x1", "x2", "f1", "f2" and of the number "12.5663706".

The size of the full source code of this program is 13 lines, including comments and empty lines. The typical size of a real-world program is between 10,000 and 10,000,000 lines of source code; some very large projects even reach 1,000,000,000 lines and above. Thus in realistic cases, there is no chance to reconstruct a useful source code from the binary code.

As a consequence, the task of software reverse engineering is extremely difficult. It is never done to save work, but only as a last resort in some special situations, for example when there are no other means to achieve interoperability. Moreover, legal protections exist for those who seek to bar reverse engineering. Thus, there is no meaningful risk of reverse engineering complex code and

any risk that does exist would not be cured by the addition of *another*[7] legal barrier.

b. Software and Business Method Innovations Would Not Benefit From The "Disclosure" Aspect of Patents.

A key benefit of the patent system is the requirement that the innovator disclose his or her invention in the patent in order to enrich the body of public knowledge and presumably enhance future innovation[8]. While this aspect of patent policy applies to hardware patents, it does not apply to software.

A typical software patent does not disclose source code. It does not even cover a complete realization of a software system, but only an elementary component of software – an algorithm. This component of the greater source code provides no significant addition to the body of

---

[7] Presumably those seeking to protect their software from reverse engineering have employed the existing legal protections afforded by the Uniform Trade Secrets Act and common law contract principles. (See e.g. California Civil Code §3426 *et seq.*). There is no reason that a miscreant bent on breaking either of these existing laws would suddenly be deterred by the added protection afforded under a new one (patent law).

[8] The disclosure required in exchange for obtaining a patent ensures that "the knowledge of the invention enures to the people" and "stimulate[s] ideas and the eventual development of further significant advances in the art." *Kewanee Oil Co.* v. *Bicron Corp.*, 416 U.S. 470, 481 (1974).

knowledge. While the source code could provide such knowledge, it commonly remains a trade secret.

Nor is it realistic to expect every created algorithm to be patented (if such protection were afforded) or that those in the software industry would expend the time to review such disclosed algorithms if they were available through the patent office. Arguably unique algorithms are so ubiquitous that the current patent system would likely be strained to the breaking point if every software programmer were to truly invest the time and resources needed to obtain patent protection for every new and innovative algorithm created.

While patents do not disclose meaningful software innovations, software programmers have themselves created a voluntary system that does. The Free and Open Source Software (FOSS)[9] community provides a working system that encourages programmers to disclose their source code. FOSS is based on copyright When a software system is released under a FOSS license, the author deliberately discloses his source code under certain conditions. In exchange he gets access to a large code base of other FOSS. That way, even direct competitors can—and in fact do—pool their efforts to create better software. Competition takes place in terms of service quality, not of the code base. To enable this system to work, the "conditions" under which source code is disclosed must be crafted very carefully. One essential condition for FOSS is that it can be copied freely by everyone who has obtained it

---

[9] For more information about free software see http://www.gnu.org.

in a legal way[10]. In particular, there must not be any control over the number of copies. As soon as there are conditions which require control over the number of copies, FOSS cannot be used.

In sum, extending patents to software would neither encourage greater disclosure nor spur greater innovation. Those interested in safe disclosure already have a ready mechanism for doing so in FOSS. Those that choose not to disclose enjoy the protections afforded by copyright, licensing, and trade secret laws.

2. Economic Imperatives And Historical Lessons Weigh Against Overly Broad Patent Monopolies.

Patents--like any other "exclusive right to [. . .] writings and discoveries" — are intended to "promote the progress of science and useful arts".[11] The patent-monopoly is the "incentive" intended to entice the progress of science and useful arts. But this economic rationale of the patent system has lead to the false conclusion that since some patents further innovation, *more* patents necessarily further more innovation. But in reality the total private commercial value of patents equals their social costs in the

---

[10] The brief submitted by *amici* Lee A. Hollar and IEEE is decidedly wrong in its characterizations of FOSS. FOSS and its incompatibility with software patents has nothing to do with any similarity to "underdeveloped economies" and has no relation to the "cloning" of others' proprietary software. Brief of Professor Lee A. Hollar and IEEE-USA as Amici Curiae at p.21, n.31.

[11] Article 1, Section 8 U. S. Constitution, § 8, cl. 8

form of an inefficient burden on a competitive market[12]. Thus, there is no "free lunch" in patenting. The "price" of granting each patent monopoly is real and should be weighed carefully.

From an economic standpoint it is essential that subject matter that is awarded patent protection be "scarce" under laissez-faire conditions, and that more of that subject matter be produced as a result of the protection awarded. But when patents are awarded for ubiquitous subject matter, observers properly complain about "trivial" grants. For an economist seeking to engineer an incentive system based on awarding limited monopolies to successful innovators, the application of such monopolies to abundant matter does not make any sense. It limits the freedom of commercial action but actually stifles the progress of science and useful arts.

a. Reliving Past Mistakes.

Until 1623, before the Statute of Monopolies was enacted in the United Kingdom, patents were granted for almost any business activity, from the right to sell textiles to the import of spices or the export of goods[13]. Patents had

---

[12] Leading economists have often taken a critical view of even the current patent system. See e.g. Hayek, F. A. v., Individualism and Economic Order, Chicago, University of Chicago Press, 1949, 114; Milton Friedman, Capitalism and Freedom, Chicago (1962), p. 127; Joseph E. Stiglitz, Making Globalization work, New York (2006), p. 103 ff.

[13] "The English Statute of Monopolies of 1623 was intended to put an end to previous abuses under which patents granted under the Royal Prerogative had sometimes protected genuine technical innovations

become a way for the government to over-regulate and thereby interfere with the market in ways unrelated to the stated goals of the patent system. About four hundred years ago, the negative effects became so unbearable that patents were limited from business activity to engineering inventions.

As society has progressed it has come full circle on the issue of patents, with some, again, seeking ever-broadening interpretations of patentable technology. In October 2003, The Federal Trade Commission issued a report entitled: To Promote Innovation: The Proper Balance of Competition and Patent Law and Policy. The Executive summary states: "A failure to strike the appropriate balance between competition and patent law and policy can harm innovation. For example, if patent law were to allow patents on 'obvious' inventions, that might have developed based on the obvious technology. [.] Conversely, competition policy can undermine the innovation that the patent system promotes if overzealous antitrust enforcement restricts the procompetitive use of a valid patent."[14]

(or useful foreign technologies newly copied and imported, which for England in the sixteenth and seventeenth centuries was far more important), but had equally been used to create private monopolies for royal favorites on such everyday items as starch, vinegar and playing cards." Christopher Wadlow, Utility and industrial applicability, in: Toshiko Takenaka (Ed.), Patent Law and Theory, A Handbook of Contemporary Research, Cheltenham, UK (2008), p. 360.

[14] FTC Report, executive summary, page 3

In joint FTC and Department of Justice hearings related the FTC report, participants "found much to praise in the current patent system",[15] but their findings indicated that "Some Modifications Are Needed to Maintain a Proper Balance of Competition and Patent Law and Policy"[16] Indeed, the summary transcripts contain a depressing, unrelenting tale of woe. In testimony after testimony, concerns were raised that the USPTO has been granting patents of questionable value. Some granted patents are considered too broad, while some are on trivialities. Still others are mistakenly granted on subject matter which is non-patentable in the first place. For many, disqualifying prior art exists, but the USPTO was plainly unaware of the existence of such art when the patent issued.

Other actors, sometimes characterized as "patent trolls", never innovate at all. Instead, they wait until others independently develop whatever technology they have patented or licensed, and then collect fees from the true innovators, who are trying to bring new products onto the market. Still other companies are accused of rushing to the patent office attempting to get a patent on whole industries, by virtue of being the first to claim an industry standard practice. Others are claiming mathematical algorithms which have always been the common heritage of all humanity. Indeed, it is not difficult to find patents that manage to suffer from all of these defects simultaneously. While such accusations have been in existence for as long

---

[15] FTC Report, executive summary, page 4

[16] FTC Report, executive summary, page 4, first conclusion

as the patent system itself,[17] there is widespread belief that the impact of non-practicing entities ("patent trolls") is much greater today than it has ever been before.

The Machine-or-Transformation test is one approach for a criterion to more clearly separate "good," permissible patents from the "bad" impermissible patents – hopefully during the patent application process long before such "bad" patents have a damaging effect on the marketplace.

b. Potential Effects of Extending Patent Protection to Software and Business Methods.

What happens if one grants exclusive rights on business or software components? Each grant of monopoly over the component part covers all uses of that component in all fields of programming or use. Since a complete software system, or a large business practice, consists of many thousands of components, there is a high risk that the software system or business practice will unknowingly violate any number of exclusive grants over various components. The result is a stifling of further innovation, an increase in transactional costs (such as insurance and litigation) for innovators, and ultimately the abandonment

---

[17] In 1895, George Baldwin Selden obtained a patent with a claim so broad that it literally encompassed most automobiles ever made—putting a gasoline engine on a carriage—despite the fact that he had never gone into production with a working model of an automobile. See Robert Patrick Merges and John Fitzgerald Duffy, Patent Law and Policy, Cases and Materials, Charlottesville (2002), pp 644-646 for more information about the Selden Patent.

of research and developments by all but a few; precisely the opposite of what the patent statutes are intended to accomplish.

To avoid this danger, software copyrights grant exclusive rights *not* on software components, but only on individual programs, i.e. realizations or expressions of software systems. Similarly, trade secrets protect the confidentiality agreed upon by the parties in which the "secret" is reposed. Neither mechanism grants a broad range of exclusivity that potentially forecloses innovation in an entire field.

An additional danger exists in granting software patents. Patent licenses typically require a fee per copy. For this reason, it is very difficult to obtain a patent license for FOSS. Accordingly, there is a long list of FOSS projects which have had to be terminated to evade the risk of a patent lawsuit.[18] These development risks are increased when the patent covers a standard file format or protocol. In such a case, even if there is a better or more efficient algorithm available than the patented one, there is no way around the patent without giving up interoperability,[19] and the patent effectively locks out FOSS from the field of programming where the standard is defined.

Software patents are in fact not patents on specific software *realizations*, but on abstract fundamental principles of programming – the *algorithms*. This brings an extremely high risk of unintentional patent infringement to

---

[18] See http://eupat.ffii.org/patents/effects/index.en.html.

[19] See the cases STAC, JPEG, MPEG, Dolby, VOIP, ASF, LZW, TTF, RSA, WWW, and RDF in footnote 11 above plus the examples in section 3.1.4 below.

software developers. A patent claim typically takes 10-100 lines to describe the algorithm in human language. Software systems consists of a large number of algorithms. A typical software system comprises 10,000 to 10,000,000 or even more lines of source code, which accounts for a minimum of 100 to 10,000 algorithms, each of which might (or might not) be covered by a patent. So instead of one patent covering one software realization, developers would be faced with a software realization covered by hundreds or thousands of patents on software *algorithms*. As a consequence, it will eventually become unfeasible to write software without infringing a large number of patents – a veritable minefield of lawsuits for even the most careful innovators.

B. A Clear Boundary Between Patentable and Non-Patentable material is Important for Continued Innovation.

To avoid the numerous dangers associated with extending the exclusive grant offered by patents to subject matter not *actually* covered by patent law it is imperative that lower courts and the Patent and Trademark Offices world-wide have the benefit of clearly defined boundaries.

1. Existing Tests For Determining the Boundaries of Patent Subject Matter.

Both Europe and the United States have heeded history's lessons and imposed limitations on patentable subject matter. The European Patent Convention, in its Art. 52(2)(c) expressly states that "schemes, rules and methods for performing mental acts, playing games or doing business, and programs for computers [...] as such [...]

shall not be regarded as [patentable] inventions". The United States, through this Court's rulings has similarly long limited the subject matter of patents to avoid abuse.

While many misquote the American rule as permitting patent protection for "anything under the sun that is made by man" the true rule states that anything made by man is patentable *except* for enumerated exceptions[20]. "The laws of nature, physical phenomena, and abstract ideas have been held not patentable." (*Id.*). It is for good reasons that, even in cases arguing for the broadest patentability, this need for limitations is endorsed.

This Court has already detailed how to translate the rules and limitations to the daily practice of granting patents. In *Gottschalk v. Benson*,[21] this Court held that a procedure for solving a given type of mathematical problem is known as an "algorithm". Allowing a patent on such a procedure would "wholly pre-empt the mathematical formula and in practical effect would be a patent on the algorithm itself". Nevertheless, the mere presence of an algorithm in a solution shall not per se exclude the possibility of a patent being granted, as long as all other requirements of patentability are met.[22] As this court held in *Diamond v. Diehr*, (occasionally "*Diehr*") any claimed algorithm has to be treated as prior art. The patent can be granted only if the same claim as the algorithm also contains some other innovation that produces a useful,

---

[20] *Diamond v. Chakrabarty,* 447 U.S. 303, 309, (1980); see also, *Diamond v. Diehr*, 450 U. S. 175 (1981).

[21] *Gottschalk v. Benson*, 409 U. S. 63 (1972)

[22] *Parker v. Flook*, 437 U.S. 584 (1978); *Diamond v. Diehr*, 450 U. S. 175 (1981)

concrete and tangible result. *Diehr* also defined the term "algorithm:"

> "1. A fixed step-by-step procedure for accomplishing a given result; usually a simplified procedure for solving a complex problem, also a full statement of a finite number of steps. 2. A defined process or set of rules that leads [sic] and assures development of a desired output from a given input. A sequence of formulas and/or algebraic/logical steps to calculate or determine a given task; processing rules."[23]

As *Diehr* noted, no distinction may be drawn between the first and second definitions. Thus, in a very real sense this Court has already spoken on the issue now before it, finding that abstractions/logic/software cannot be claimed in a patent, absent the addition of other patentable subject matter within the claim.

Indeed, this Court has been clear and consistent in its pronouncements. It is the Federal Circuit Courts of Appeal that have been inconsistent. The United States Court of Appeals for the Federal Circuit in *In re Hiroyuki IWAHASHI*,[24] provides one such diversion from existing precedent:

---

[23] Brief for Petitioner in *Diamond v. Bradley*, O.T. 1980, No. 79-855, p. 6, n. 12, quoting C. Sippl & R. Sippl, Computer Dictionary and Handbook 23 (2d ed. 1972).

[24] *In re Hiroyuki IWAHASHI, Yoshiki Nishioka and Mitsuhiro Hakaridani*, 12 U. S. P. Q. 2d 1908; 888 F.2d 1370 (1989)

> Over-concentration on the word "algorithm" alone, for example, may mislead. The Supreme Court carefully supplied a definition of the particular algorithm before it [in Benson], i.e., "[a] procedure for solving a given type of mathematical problem." The broader definition of algorithm is "a step-by-step procedure for solving a problem or accomplishing some end." Webster's New Collegiate Dictionary (1976).

The Court of Appeals thereby implied that there would exist mathematical, abstract algorithms as opposed to (non-mathematical, non-abstract) "applied" algorithms. This error was then reiterated and wholeheartedly adopted by the Circuit Court in *State Street*.[25]

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

---

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

The *State Street*[26] decision was contrary to a broad range of authorities including the holdings of *Gottschalk, Parker*[27], and *Diehr*. *State Street's* pronouncement that an algorithm that produces a particular result is protectable is directly contrary to the teachings of *Diehr*:

> [.] the algorithm is treated for 101 purposes as though it were a familiar part of the prior art; the claim is then examined to determine whether it discloses "some other inventive concept."

In other words, under *Diehr,* it is plain that the calculation itself *cannot* be claimed. It must be effectively discarded and the remaining claim terms must be analyzed in order to determine whether "some other inventive concept" exists, which can then be afforded patent protection.

Summarizing, the case law of the Supreme Court defined a clear line, in which algorithms could not be claimed on their own and had to be accompanied – in the same claim – by something else meeting the requirements of patentability. This is not unlike the decisions of the Board of Appeals of the European Patent Office, which

---

[26] As discussed below, *State Street's* damage did not stop at the water's edge. Seeking to harmonize its own laws with those of the United States, the European Patent Office has also relied upon *State Street's* broad, and ultimately misguided reading of U.S. patent law.

[27] 437 U.S. 584 (1978)

interpreted the so-called "technology requirement" in an analogous way.[28]

a. The European Approach: Art 52 EPC and Technicity

European law provides for an explicit exclusion of patents on software and business methods. Art 52 of the European Patent Convention (EPC) reads:

> "(2) The following in particular shall not be regarded as [patentable] inventions L...1:
>
> (c) schemes, rules and methods for performing mental acts, playing games or doing business, and programs for computers; [...]
>
> (3) The provisions of paragraph 2 shall exclude patentability of the subject-matter or activities referred to in that provision only to the extent to which a European patent application or European patent relates to such subject- matter or activities as such."

In light of Art 52(2)(c) of the EPC it may seem counterintuitive that any software patents would issue in

---

[28] The Board of Appeals of the European Patent office only briefly diverged from this line of reasoning when it sought to harmonize its practice with the United States after *State Street Bank & Trust Co. v. Signature Financial Group Inc.,* 149 F. 3d 1368 (Fed. Cir. 1998). As discussed below, that issue has since been resolved by the European Parlament.

Europe. Yet, such patents have issued, calling into question the validity of those issued patents. This occurred because from 1998 onward, the European Patent Office changed its patent granting practice for software without regard to Art 52 (2) (3). Specifically, prior to 1998 – not unlike the United States prior to *State Street* – algorithms could not be claimed easily in Europe. But after 1998, the European Patent Office switched to an interpretation that left little to no room for the application for the exclusion of software found in Article 52.

As tension increased with respect to the European Patent Office's treatment of software patents, intense lobbying for a European Directive that would have effectively abrogated[29] Article 52 was commenced, most importantly by the European Patent Office itself. The European Commission then issued the proposal COM (2002) 92 final 2002/0047 (COD), 2002 which, if approved, would have effectively "cured" most of the dubious patents. However, the proposal was definitively refused on July 6,

---

[29] Page 7 of the Directive proposal states that computer programs which have "a technical character [...] are not considered to fall under the exclusion in Article 52(2) as they are considered not to relate to programs for computers 'as such'." On the same page we find that "all programs when run in a computer are by definition technical (because a computer is a machine) [...]." In other words: all software would be technical, thus the exclusion by Art. 52(2) EPC would be void. This Directive proposal was rejected by the European Parliament on 6th July 2005 and was never implemented.

2005 by the European Parliament. Thus, currently, the prohibition against such patents found in Article 52 remains good law.

b. The Lower Court's Machine-or-Transformation Test

According to the Machine-or-Transformation patent eligibility test, a claim to a process qualifies to be considered for patenting only if it (1) is implemented with a particular machine, that is, one specifically devised and adapted to carry out the process in a way that is not concededly conventional and is not trivial; or else (2) transforms an article from one thing or state to another.

This test is a means to an end. It is there not to cross, but to define the border between patentable new and useful processes and non-patentable laws of nature, physical phenomena, and abstract ideas. This answers the first question presented by the petitioners.

Regarding the second question, the American Inventor Protection Act in 1999 introduced a limited "prior use" defense in American patent law, specifically for "methods of doing or conducting business". This regulation, inserted as § 273 into the American Patent Act, implicitly acknowledges the existence of business methods, by virtue of its limitation to those methods.

Unlike the second question of the petitioners suggests, 35 U. S. C. § 273 does not reflect a "clear Congressional intent that patents protect 'method[s] of doing or conducting business.'" The regulation only recognizes business method patents as a fact and tries to bring their economic impact under control by setting a limit.

This is achieved by the Machine-or-Transformation test: setting a limit. It does not contradict 35 U. S. C. § 273, but complements it.

On the other hand one must take very seriously the concerns raised by Judge Mayer in his dissent to this test.

> "Bilski, for example, could simply add a requirement that a commodity consumer install a meter to record commodity consumption. He could then argue that installation of this meter was a 'physical transformation'."

The same kind of "clever draftsmanship" has been advocated in Europe by the EPO, which for many years saw its job as one of "helping its clients, the would- be patent holders, apply for and *receive* patents" and featured on its website a document which described in detail how to circumvent Art. 52(2) EPC.

Neither the EPO, nor the PTO, nor the Federal Circuit Courts should be in the business of "helping" applicants gain patent protection. As discussed in the earlier portion of this brief, there are very real cost to society for each patent that issues. Those costs reach unacceptably high levels when the patent has the effect of excluding others from participation in abstract ideas, laws of nature, physical phenomena and the like[30].

---

[30] *Diamond v. Chakrabarty,* 447 U.S. 303, 309, (1980); see also, *Diamond v. Diehr*, 450 U. S. 175 (1981).

2. Improving the Patentability Tests

The purpose of the Machine-or-Transformation test, reiterated in *Bilski*, is to limit patentability in a manner consistent with existing Supreme Court precedent and to exclude undesired patent applications. To serve this purpose, *Amici* propose that the Machine-or-Transformation test be supplemented with the following measures:

1. The Machine-or-Transformation test must be applied to the claimed object rather than to the patent claim as a whole. An object can only be claimed in a patent if it constitutes a new, concrete realization of a machine or transformation.

   Without these further requirements, patent claims may incidentally involve machines or transformations which are irrelevant to the claim, while still effectively working to impermissibly patent laws of nature, physical phenomenon or abstract ideas.

2. In applying the "suggestion test", an ability to combine or modify prior art references should be assumed that is consistent with ordinary creativity and problem-solving skills in the art.

   The suggestion test states that if the prior art would have already suggested the claimed invention, then the claimed invention is

obvious. However, the participants at the FTC/DOJ Hearings expressed concern with some recent applications of the suggestion test. To show that a claimed invention is obvious, the USPTO had to point to particular items of prior art that concretely suggest how to combine all of the features of a claimed invention. It is those fields where the combining of elements is most straightforward that the chances are least that one can find an existing document that outlines that particular combination, simply because nobody having ordinary skill in the art would ever require instruction at that level. How to combine existing elements would be considered "too obvious" to need mention.

3. An expanded version of the suggestion test is relevant to all inventions that use a computer. Simply writing a computer program and then running it on a computer is not novel, under the meaning of patent law. And doing so is obvious, again under the meaning of patent law.

   These days, we experience a similar dawn of mankind as Henry Ford experienced 100 years ago.[31] Back then it was completely obvious for an engineer to put an engine on a carriage. In these days, it is completely obvious for

[31] *See also* the FTC report, executive summary, page 12

someone skilled in the art of computer programming to convert mathematical principles into computer programs. Thus while what *Bilski* has done may be termed "innovative" in an entrepreneurial sense, it is in no way innovative in the sense of patent law

4. The "commercial success test" should be expressly abandoned. This test states that in some circumstances, courts may consider the commercial success of a claimed invention to indicate that it was not obvious. It must have satisfied an unmet need, which was not obvious to anybody else, or else they would have already been satisfying it.

> Commercial success comes from many factors, many of which have nothing to do with the claimed invention, e.g. Marketing, advertising, a pre-existent dominant position, etc. Henry Ford, after all, claimed that his business success was based on the completely obvious idea of putting a gasoline engine on a chassis. His great success arose from other factors.

These minor modifications or clarifications to the existing test will only serve to better refine the existing law to ensure that patent protection continues to serve its intended purposes while excluding undesirable applications

for material that ultimately is not entitled to patent protection.

V.

CONCLUSION

For the foregoing reasons *Amici* respectfully urge this court to re-affirm its long-standing rules prohibiting expansion of patent protection to subject matter that has the effect of precluding innovators from employing abstract

ideas, algorithms, laws of nature and the like. The Federal Circuit's Machine-or-Transformation test, in particular when slightly modified as suggested herein, provides the clarity needed with respect to business method and software patents. As such, this Court should Affirm.

Dated: October 2, 2009

COUNSEL OF RECORD
Allonn E. Levy
Hopkins & Carley
408.286.9800 (main)
70 South First Street
San Jose, CA 95113

As Counsel *Amici Curiae:*

Foundation for a Free
Information Infrastructure,

IP Justice,

Dr. rer. nat. Peter Gerwinski ,
Laura Creighton,
Mag. iur. Georg Jakob,
Dipl. Kfm. & M.A. André
Rebentisch