SOFTWARE PATENTING AND SECTION 101'S GATEKEEPING FUNCTION

Andrew Chin[*]

[DRAFT — 7/26/2019]

INTRODUCTION

Software-related inventions have had an uneasy relationship with the patent-eligible subject matter requirement of Section 101 of the Patent Act.[1] In applying the requirement, the Supreme Court has historically characterized mathematical algorithms and formulas *simpliciter* as sufficiently analogous to laws of nature to warrant judicial exclusion as abstract ideas.[2] The Court has also found "the mere recitation of a generic computer" in a patent claim as tantamount to "adding the words 'apply it with a computer,'" a mere drafting effort that does not relieve "the pre-emption concern that undergirds our § 101 jurisprudence."[3] Lower courts, patent counsel and commentators have struggled to apply these broad principles to specific software-related inventions, a difficulty largely rooted in the many forms and levels of abstraction in which mathematical algorithms can be situated, both in the computing context and in the terms of a patent claim.[4] Consequently, widely varying approaches to claiming inventions that involve algorithms in their use have perennially complicated efforts to develop a coherent doctrine of unpatentable abstract ideas.

---

* Paul B. Eaton Distinguished Professor of Law, University of North Carolina; J.D., Yale Law School; D.Phil. (Mathematics), University of Oxford.

[1] *See* 35 U.S.C. § 101 ("Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.").

[2] *See* Parker v. Flook, 437 U.S. 584, 590 (1978) (citing Gottschalk v. Benson, 409 U.S. 409 U.S. 63, 67 (1972)) ("Reasoning that an algorithm, or mathematical formula, is like a law of nature, *Benson* applied the established rule that a law of nature cannot be the subject of a patent.").

[3] *See* Alice Corp. v. CLS Bank Int'l, 134 S.Ct. 2358 (2014) (citing Mayo Collab. Svcs. v. Prometheus Labs., 132 S.Ct. 1289, 1294-1301 (2012)).

[4] *See, e.g.,* Jeffrey A. Lefstin, *The Three Faces of Prometheus: A Post-Alice Jurisprudence of Abstractions*, 16 N.C. J. L. & TECH. 688 (2015) ("The pivotal question … perhaps for software patents more generally, is whether specific information-processing techniques are abstract ideas."); *see generally* JAMES BESSEN & MICHAEL J. MEURER, PATENT FAILURE: HOW JUDGES, BUREAUCRATS AND LAWYERS PUT INNOVATORS AT RISK 201 (2008) (arguing that "the abstractness of software technology inherently makes it more difficult to place limits on abstract claims in software patents").

In the computing context, the term "algorithm" can refer to any "finite sequence of steps" that accomplishes a given task.[5] As an algorithm is usually described in the computer science literature, it is common for some or all of the "steps" themselves to be tasks that can be decomposed further into sequences of more basic steps. A computer system thereby typically involves numerous "abstraction layers," with each successive, more abstract, layer implementing its own set of functions through various algorithms comprising sequences of functions previously implemented by the more concrete layers below.[6] To make matters even more complicated, abstraction layers often provide multiple distinct implementations and interpretations of a single function, using a versatile programming technique known as "indirection."[7] For example, the FreeBSD operating system uses indirection to implement a single "read system call" operation on disparate filesystem organizations such as those in PC hard drives, CD-ROMs, and USB sticks.[8]

As of this writing, the Senate Judiciary Subcommittee on Courts, Intellectual Property and the Internet is considering draft legislation to overhaul existing law relating to patent-eligible subject matter, *inter alia*, by specifying that (1) "the provisions of section 101 shall be construed in favor of eligibility," (2) "no implicit or other judicially created exceptions to subject matter eligibility … shall be used to determine patent eligibility under section 101, and all cases establishing or interpreting those exceptions to eligibility are hereby abrogated," and (3) "eligibility … under section 101 shall be determined without regard to … any other considerations relating to sections 102, 103, or 112."[9] According to the bill's drafters, the new statute codifies the principle that "statutory

---

[5] *See* MICROSOFT COMPUTER DICTIONARY 19 (1999) (defining "algorithm" as "[a] finite sequence of steps for solving a logical or mathematical problem or performing a task").

[6] *See* ANDREW S. TANENBAUM, STRUCTURED COMPUTER ORGANIZATION (1979).

[7] *See* Diomidis Spinellis, *Another Level of Indirection*, *in* BEAUTIFUL CODE: LEADING PROGRAMMERS EXPLAIN HOW THEY THINK 279–291 (Andy Oram & Greg Wilson, eds. 2007). Indirection is such a versatile approach to abstracting away implementation details that the claim that "[a]ll problems in computer science can be solved with another layer of indirection" has become a well-known aphorism among programmers. *See id.* at 279.

[8] *See id.* at 279-82.

[9] *See* Thom Tillis, *Sens. Tillis and Coons and Reps. Collins, Johnson, and Stivers Release Draft Bill Text to Reform Section 101 of the Patent Act*, Press Release (May 22, 2019), https://www.tillis.senate.gov/2019/5/sens-tillis-and-coons-and-reps-collins-johnson-and-stivers-release-draft-bill-text-to-reform-section-101-of-the-patent-act (hereinafter "Draft Bill Text").

exceptions should be the only basis for excluding inventions from eligibility and courts may not expand them."[10] The text of the proposed statute, however, simply recites the already existing categories of statutory subject matter (process, machine, manufacture, composition of matter, improvement) without any mention of exceptions while specifying that patentable utility requires "specific and practical utility in any field of technology through human intervention."[11]

As the judicially created exceptions from patent-eligible subject matter hang in the balance, it is a critical time to examine the form and function of the courts' patent-eligibility jurisprudence to date, particularly in the software field. This chapter identifies and reviews three conceptually divergent judicial approaches to the patent-eligibilty of software-related inventions.

Part I of this chapter examines courts' efforts in past decades to ground the eligibility of some software-related inventions in the statutory category of "new and useful … machine[s]."[12] This approach was problematic insofar as it tended to obscure considerations of the underlying mathematical algorithm in other aspects of the patentability analysis. The proposed legislation would likely send courts down this road again, in that software-related inventions would fall under the "process" and "machine" statutory categories, with a general-purpose computer programmed to perform any practical function being eligible as a "machine."

Part II describes and critiques the current framing of preemption as the central concern necessitating the judicial exclusion of certain software-related inventions. This preemption concern neither accurately captures the rationale for judicial exclusion nor provides adequate guidance regarding the eligibility of software-related claims.

Part III highlights the judicial exclusions' historic and enduring role in obviating other patentability inquiries that would be inapposite as applied to the claimed subject matter. This gatekeeping function represents an independently sufficient, jurisprudential, rationale for the patent-eligible subject matter requirement and provides a precise criterion by which examiners and courts can distinguish between abstract ideas and their practical applications in the field of computing.

---

[10] *See* Thom Tillis, *Tillis, Coons Vet Patent Eligibility Bill Principles with Stakeholders*, Press Release (Mar. 27, 2019), https://www.tillis.senate.gov/2019/3/tillis-coons-vet-patent-eligibility-bill-principles-with-stakeholders.

[11] *See* Tillis, Draft Bill Text, *supra* note 9.

[12] *See* 35 U.S.C. § 101.