Patent Eligibility of Programming Languages and Tools

Sebastian Zimmeck*

While patent eligibility is a frequently discussed topic for computer programs in general, it is less often addressed for the instruments that are used for creating those programs—programming languages and tools. However, the patent eligibility analysis for programming languages and tools is particularly important because it determines whether standards for creating computer programs can be made proprietary under the patent laws subjecting them to the exclusive control of the patent holders. Therefore, it is the goal of this Article to provide a basic analysis for patent eligibility of programming languages and their implementations in programming tools.

This Article presents an attempt to carefully refine the law of patent eligibility by applying the existing statutory framework and judicial precedents to the field of programming languages and tools. The analysis will be structured into five Parts. It will begin with a brief inquiry into the relationship between patent and copyright eligibility (Part I). Then the law of patent eligibility is surveyed (Part II) and applied to programming languages and tools (Part III). This application of the law is followed by an analysis of a possible First Amendment limitation (Part IV) and complemented by a final summary of the results (Part V).

* © 2010 Sebastian Zimmeck. Dr. iur., Christian-Albrechts-University at Kiel School of Law, 2008; LL.M., University of California, Berkeley, 2006; First State Examination, Christian-Albrechts-University at Kiel School of Law, 2003.

I. RELATIONSHIP OF PATENT TO COPYRIGHT ELIGIBILITY

Programming languages unite both expressive and functional aspects of intellectual property law.[1] Emphasizing the functional aspect, some commentators argue that the patent law regime is a suitable fit for protection of programming languages.[2] According to this opinion, a conflict between copyright and patent protection would not be possible. However, there is some argument for copyright protection of programming languages as well.[3] Further, programming tools, referring to computer programs for facilitating the creation of other computer programs, are also categorically covered by copyright law.[4] Thus, assuming copyright eligibility of programming languages and tools could lead to cumulative application of copyright and patent law, resulting in frictions between the two legal regimes.

However, patent law relates to the implementation of ideas, while copyright law relates to their expression.[5] Therefore, while the copyright analysis of programming languages refers to the expressive form of the language, the patent analysis is aimed at the language's underlying function. Copyright analysis focuses on the lexical, syntactic, and semantic structures of a programming language without actually considering what they accomplish. Copyright analysis of programming tools is also limited to expressive form, while the patent analysis refers to functionality. Accordingly, both legal regimes coexist and do not mutually exclude one another.[6] A finding of whether programming

1. In this Article, "programming language" refers to a high-level programming language that is independent from any target machine language.

2. *See, e.g.*, Marci A. Hamilton & Ted Sabety, *Computer Science Concepts in Copyright Cases: The Path to a Coherent Law*, 10 HARV. J.L. & TECH. 239, 263 n.95 (1997); Elizabeth G. Lowry, Comment, *Copyright Protection for Computer Languages: Creative Incentive or Technological Threat?*, 39 EMORY L.J. 1293, 1297 (1990); Richard H. Stern, *Copyright in Computer Programming Languages*, 17 RUTGERS COMPUTER & TECH. L.J. 321, 378 (1991).

3. *See* Lotus Dev. Corp. v. Paperback Software Int'l, 740 F. Supp. 37, 72 (D. Mass. 1990) (in dictum) ("[D]efendants, . . . have cited no precedent that supports the contention that a 'language' . . . is not copyrightable."); *see also* Michael P. Doerr, *Java: An Innovation in Software Development and a Dilemma in Copyright Law*, 7 J. INTELL. PROP. L. 127, 146 (1999) ("A court may find the high-level component of the Java language uncopyrightable . . . . However, the Java bytecodes seem to be copyrightable subject matter. . . .").

4. 17 U.S.C. § 101 (2006) (defining a computer program as "a set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result").

5. *See* Brian F. Fitzgerald, *Software as Discourse: The Power of Intellectual Property in Digital Architecture*, 18 CARDOZO ARTS & ENT. L.J. 337, 359 (2000); Dennis S. Karjala, *Distinguishing Patent and Copyright Subject Matter*, 35 CONN. L. REV. 439, 448 (2003).

6. STEVEN W. LUNDBERG, STEPHEN C. DURANT & ANN M. MCCRACKIN, ELECTRONIC AND SOFTWARE PATENTS: LAW AND PRACTICE § 2.05.A (2005).

languages or tools are copyright eligible has no bearing on the determination of patent eligibility.

II. SURVEY OF THE LAW ON PATENT ELIGIBILITY

Generally, "anything under the sun that is made by man" is patent-eligible.[7] However, statutory law further qualifies this broad statement requiring patent-eligible subject matter to be an "invention or discovery of a new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof."[8] Thus, only four subject matters are patent-eligible: processes, machines, manufactures, and compositions of matter.[9] The relationship between these four patent-eligible subject matters can be described as follows: Processes are exercises of technological skill, manufactures and compositions of matter are products of that skill, and machines are the tools through which that skill is exercised.[10] In addition, improvements of the four types of patentable subject matters are also patent-eligible.[11]

The requirement that only new and useful subject matter is patent-eligible ensures that new inventions can employ fundamental principles, such as abstract ideas, laws of nature, and natural phenomenon, without claiming them in the abstract.[12] Fundamental principles alone are not patent-eligible.[13] This is even true if they are limited to a particular practical application.[14] However, these fundamental principles can be applied in new and useful ways to patent-eligible processes, machine, compositions of matter, manufactures, or improvements.[15]

Having set the stage, this Part will now present a survey of the law on patent eligibility. Specifically, it will discuss the meaning of the terms

7. Diamond v. Chakrabarty, 447 U.S. 303, 309 (1980) (quoting S. Rep. No. 82-1979, at 5 (1952)). "Patent eligibility" is also referred to as "patentability" or "statutory subject matter."

8. 35 U.S.C. § 101 (2006).

9. *In re* Nuijten, 500 F.3d 1346, 1354 (Fed. Cir. 2007).

10. *Id.* at 1352.

11. 35 U.S.C. § 101.

12. *Id.*

13. *E.g.*, Le Roy v. Tatham, 55 U.S. 156, 175 (1853); Gottschalk v. Benson, 409 U.S. 63, 67 (1972).

14. *E.g.*, Diamond v. Diehr, 450 U.S. 175, 192 n.14 (1981); Parker v. Flook, 437 U.S. 584, 590 (1978).

15. *Cf.* Rubber-Tip Pencil Co. v. Howard, 87 U.S. 498, 507 (1874) ("An idea of itself is not patentable, but a new device by which it may be made practically useful is."); *In re* Alappat, 33 F.3d 1526, 1543 (Fed. Cir. 1994) ("[T]he dispositive inquiry is whether the claim as a whole is directed to statutory subject matter, it is irrelevant that a claim may contain, as part of the whole, subject matter which would not be patentable by itself."); Note, *Pure Fiction: The Attempt To Patent Plot*, 19 HARV. J.L. & TECH. 231, 244 (2005) ("[Patent law] is devoted to the protection of nonabstract ideas with practical applications.").

"machine," "process," "manufacture," and "new and useful." This Article will not address compositions of matter and improvements.

A. Machine

Traditionally, the United States Supreme Court defined a "machine" as a "concrete thing, consisting of parts or of certain devices and combination of devices."[16] The definition "includes every mechanical device or combination of mechanical powers and devices to perform some function and produce a certain effect or result."[17] For purposes of distinguishing machines from processes, the Supreme Court qualified a machine as a thing "visible to the eye, an object of perpetual observation," while a process was defined as "an act or a mode of acting. . . . a conception of the mind, seen only by its effects when being executed or performed."[18] Consequently, according to the traditional definition, a machine is characterized by its physical nature and performance of a function.

In *In re Alappat*, the United States Court of Appeals for the Federal Circuit extended the machine definition into the intangible realm, stating, "programming creates a new machine."[19] Addressing machine claims in a means-plus-function format, the court found that the programming of a general-purpose computer was determinative for patent eligibility. The court reasoned that the programming can transform a general-purpose computer into a special-purpose computer. A machine programmed in a new and unobvious way is physically different from a machine without that program.[20] *Alappat* therefore stands for the proposition that the programming of a computer can be the essential element for creation of a machine.

Professor Pamela Samuelson and other commentators went beyond *Alappat*, concluding that computer programs *are*, in fact, machines.[21]

---

16. Burr v. Duryee, 68 U.S. 531, 570 (1863).

17. Corning v. Burden, 56 U.S. 252, 267 (1854); SiRF Tech. v. Int'l Trade Comm'n, 601 F.3d 1319, 1332 (Fed. Cir. 2010); *In re* Ferguson, 558 F.3d 1359, 1364 (Fed. Cir. 2009); *In re* Nuijten, 500 F.3d 1346, 1356 (Fed. Cir. 2007).

18. *See* Expanded Metal Co. v. Bradford, 214 U.S. 366, 384 (1909).

19. 33 F.3d at 1545 (citing *In re* Freeman, 573 F.2d 1237, 1247 n.11 (C.C.P.A. 1978).

20. WMS Gaming, Inc. v. Int'l Game Tech., 184 F.3d 1339, 1349 (Fed. Cir. 1999); *In re* Lowry 32 F.3d 1579, 1583 (Fed. Cir. 1994); *In re* Bernhart, 417 F.2d 1395, 1400 (C.C.P.A. 1969).

21. Pamela Samuelson et al., *A Manifesto Concerning the Legal Protection of Computer Programs*, 94 COLUM. L. REV. 2308, 2320 (1994); *see also* THOMAS H. CORMEN ET AL., INTRODUCTION TO ALGORITHMS 13 (3d ed. 2009) ("[W]e should consider algorithms, like computer hardware, as a technology."); John A. Gibby, *Software Patent Developments: A Programmer's Perspective*, 23 RUTGERS COMPUTER & TECH. L.J. 293, 345 (1997) ("[P]atent law

Programs are entities constructed in the medium of text that bring about useful results.[22] Categorizing programs as machines is not an abstract metaphorical statement, but rather expresses that programs are actual and real machines in the terminology of section 101 of the Patent Act.[23] While some machines are built from physical structures like gears, wires, and screws, programs are built from information structures, such as algorithms and data structures.[24] Thus, writing programs is an industrial design process akin to the design of physical machines.[25] For this reason, programs are rightly categorized as machines under section 101.

B. Process

While a machine is the result of invention, a process is the result of discovery.[26] Process is statutorily defined as "process, art, or method, and includes a new use of a known process, machine, manufacture, composition of matter, or material."[27] Consequently, the definitions of machine, manufacture, and composition of matter can become relevant for the process analysis as well. In this regard, it is irrelevant whether the employed machine, manufacture, or composition of matter is already in the prior art. A process making a new use of a prior art machine, for example, can still be patent-eligible.[28] In such case, the patent eligibility of the process does not depend on the characteristic principle of the machine, even if it may be essential to the process.

The first element of a process is an act or a series of acts.[29] This element requires that certain things be done with certain substances and,

should view a program or data structure as an independent machine just as it does with mechanical inventions.").

22. Samuelson et al., *supra* note 21, at 2320.

23. *Id.*

24. *Id.* at 2321.

25. *Id.* at 2327.

26. Corning v. Burden 56 U.S. 252, 267 (1854). The term "discovery" is not used uniformly. It can refer to patent-eligible subject matter. *See* U.S. CONST. art. 1, § 8, cl. 8. However, discovery can also refer to a patent-ineligible law of nature or other fundamental principle. *See, e.g.*, *In re* Alappat, 33 F.3d 1526, 1582 (Fed. Cir. 1994) (Rader, J., concurring).

27. 35 U.S.C. § 100(b) (2006). Before the enactment of the Patent Act of 1952, a "process" was commonly referred to as an "art." *See, e.g.*, Pasquale J. Federico, *Commentary on the New Patent Act*, 35 U.S.C.A. 1 (1954), *reprinted in* 75 J. PAT. & TRADEMARK OFF. SOC'Y 161, 176 (1993).

28. *See* Diamond v. Diehr, 450 U.S. 175, 183-84 (1981); Cochrane v. Deener, 94 U.S. 780, 788 (1876); Federico, *supra* note 27, at 178.

29. *See, e.g.*, Gottschalk v. Benson, 409 U.S. 63, 70 (1972); *Deener*, 94 U.S. at 788; *In re* Ferguson, 558 F.3d 1359, 1364 (Fed. Cir. 2009); *In re* Nuijten, 500 F.3d 1346, 1355 (Fed. Cir. 2007); *see also In re* Kollar, 286 F.3d 1326, 1332 (Fed. Cir. 2002) (requiring a series of acts thereby excluding single acts from patent-eligible processes).

in the case of more than one act, in a certain order.[30] After identifying acts or a series of acts, a useful and important clue for determining whether those acts or series of acts are sufficient for being patent-eligible processes is provided by the two-prong machine-or-transformation test.[31] Under the test, subject matter is patent-eligible as a process under section 101 if (1) it is tied to a particular machine or apparatus or (2) transforms a particular article into a different state or thing.[32] Justice Stevens specified in his concurring opinion in *Bilski v. Kappos* that the transformation prong of the test requires a physical transformation of an article to a different state or thing.[33] However, even if the machine-or-transformation test is not satisfied, it cannot be concluded that the claimed subject matter is patent-ineligible.[34]

The Federal Circuit also employs the "mental steps" doctrine as a test for processes.[35] Originally developed by the Court of Customs and Patent Appeals, the mental steps doctrine is interpreted by the Federal Circuit to exclude from patent eligibility stand-alone processes of human thinking.[36] For example, a process for arbitration resolution itself is not patent-eligible.[37] Beyond the mental steps doctrine, earlier tests considered by the courts to exclude certain subject matter from patent

30. *See, e.g.*, *Diehr*, 450 U.S. at 183-84; Expanded Metal Co. v. Bradford, 214 U.S. 366, 384 (1909); New Process Fermentation Co. v. Maus, 122 U.S. 413, 428 (1887); *Deener*, 94 U.S. at 787.

31. *See* Bilski v. Kappos, 130 S. Ct. 3218, 3227 (2010); Gottschalk v. Benson, 409 U.S. 63, 70 (1972).

32. *See Diehr*, 450 U.S. at 182-83; Parker v. Flook, 437 U.S. 584, 589 n.9 (1978); *Benson*, 409 U.S. at 71; *Deener*, 94 U.S. at 788; SiRF Tech., Inc. v. Int'l Trade Comm'n, 601 F.3d 1319, 1332 (Fed. Cir. 2010); Prometheus Labs., Inc. v. Mayo Collaborative Servs., 581 F.3d 1336, 1342 (Fed. Cir. 2009); *In re Ferguson*, 558 F.3d at 1363; *In re* Bilski, 545 F.3d 943, 954 (Fed. Cir. 2008). For the first alternative of the machine-or-transformation test, the machine definition becomes relevant.

33. *See Bilski*, 130 S. Ct. at 3246 ("[W]e consistently focused the inquiry on whether an "art" was connected to a machine or *physical* transformation . . . ." (emphasis added)) (citing *Expanded Metal*, 214 U.S. at 383); The Telephone Cases, 126 U.S. 1, 533-37 (1888); *Deener*, 94 U.S. at 788-88; Corning v. Burden, 56 U.S. 252, 267-68 (1854).

34. *Bilski*, 130 S. Ct. at 3227; *Flook*, 437 U.S. at 589 n.9; *Benson*, 409 U.S. at 71. *But see In re Bilski*, 545 F.3d at 954 ("The Supreme Court, however, has enunciated a definitive test to determine whether a process claim is tailored narrowly enough to encompass only a particular application of a fundamental principle rather than to pre-empt the principle itself.").

35. *See In re* Comiskey, 554 F.3d 967, 979-80 (Fed. Cir. 2009).

36. *See In re* Abrams, 89 U.S.P.Q. (BNA) 266, 270 (C.C.P.A. 1951); *In re* Yuan, 89 U.S.P.Q. (BNA) 324, 329 (C.C.P.A. 1951); *In re* Heritage, 32 C.C.P.A. 1170, 1174 (C.C.P.A. 1945). *See generally* Katharine P. Ambrose, Comment, *The Mental Steps Doctrine*, 48 TENN. L. REV. 903 (1981); Andrew W. Torrance, *Neurobiology and Patenting Thought*, 50 IDEA 27 (2009).

37. *See Comiskey*, 554 F.3d at 980; *see also* Greenewalt v. Stanley Co. of Am., 54 F.2d 195, 196 (3d Cir. 1931) ("We do not find authority in the law for the issuance of a patent for results dependent upon such intangible, illusory, and nonmaterial things as emotional or aesthetic reactions.").

eligibility included the technological arts test,[38] the Freeman-Walter-Abele test,[39] and the "physical steps" test,[40] the latter two of which were replaced by the "useful, concrete, and tangible result" test,[41] which in turn was put to rest in *Bilski v. Kappos*.[42] Only the technological arts test did not require patent-eligible subject matter to have some physical or tangible characteristics.[43]

C. Manufacture

The product of a process or a machine can be a manufacture.[44] The Supreme Court read the term "manufacture" in section 101 to mean "the production of articles for use from raw or prepared materials by giving to these materials new forms, qualities, properties, or combinations, whether by hand-labor or machinery."[45] For an article to be a manufacture there must be a transformation of material and a new and different article must emerge "having a distinctive name, character, or use."[46] Although the Federal Circuit has followed Supreme Court precedents, it has interpreted the rule to exclude intangible subject matter from being a manufacture.[47]

38. *In re* Musgrave, 431 F.2d 882, 893 (C.C.P.A. 1970).

39. *See In re* Abele, 684 F.2d 902 (C.C.P.A. 1982); *In re* Walter, 618 F.2d 758 (C.C.P.A. 1980); *In re* Freeman, 573 F.2d 1237 (C.C.P.A. 1978).

40. *In re* Grams, 888 F.2d 835, 838-39 (Fed. Cir. 1989); *see also In re* Schrader, 22 F.3d 290, 294 (Fed. Cir. 1994).

41. State St. Bank & Trust Co. v. Signature Fin. Group, Inc., 149 F.3d 1368, 1374 (Fed. Cir. 1998) (citing *In re* Alappat, 33 F.3d 1526, 1557 (Fed. Cir. 1994)); *see also* AT&T Corp. v. Excel Commc'ns, Inc., 172 F.3d 1352, 1359 (Fed. Cir. 1999).

42. *See* Bilski v. Kappos, 130 S. Ct. 3218, 3259 (2010) (Breyer, J., concurring) (citing *State St. Bank & Trust Co.*, 149 F.3d at 1373).

43. *See In re* Musgrave, 431 F.2d 882, 894 (C.C.P.A. 1970); *see also In re* Toma, 575 F.2d 872, 877-78 (C.C.P.A. 1978). *But see Ex parte* Lundgren, 76 U.S.P.Q.2d (BNA) 1385, at *11 (Bd. Pat. App. & Int. 2005) (determining that there is no judicially recognized "technological arts" test under section 101). For a more detailed history of the mental steps test, the Freeman-Walter-Abele test, the physical steps test, and the technological arts test, see Donald S. Chisum, *The Future of Software Protection: The Patentability of Algorithms*, 47 U. PITT. L. REV. 959, 970 (1986).

44. *See* Am. Wood-Paper Co. v. Fibre Disintegrating Co., 90 U.S. 566, 595 (1874).

45. Diamond v. Chakrabarty 447 U.S. 303, 308 (1980) (quoting Am. Fruit Growers, Inc. v. Brogdex Co., 283 U.S. 1, 11 (1931)).

46. Hartranft v. Wiegmann, 121 U.S. 609, 615 (1887).

47. *See In re* Nuijten, 500 F.3d 1346, 1356 (Fed. Cir. 2007); Bayer AG v. Housey Pharm., Inc., 340 F.3d 1367, 1371 (Fed. Cir. 2003) (construing the term "product" in section 271(g) according to "manufacture" in section 101); NTP, Inc. v. Research in Motion, Ltd., 418 F.3d 1282, 1323 (Fed. Cir. 2005); *see also* 1 DONALD S. CHISUM, CHISUM ON PATENTS ch. 1.02[3][c] (2009).

D. *New and Useful*

According to section 101, patent eligibility requires subject matter to be "new and useful."[48] The classification of subject matter as "new" is related to the novelty analysis under section 102.[49] In some cases, the Court of Customs and Patent Appeals equated both analyses.[50] The court reasoned that it was Congress's intent to let courts evaluate the novelty of an invention under section 102 simply by naming the requirement first in section 101.[51] According to this view, the criteria for determining whether a given subject matter is new within the meaning of section 101 is no different than the criteria for determining whether the subject matter possesses novelty under section 102. Thus, that which possesses statutory novelty under section 102 is also new within the meaning of section 101.[52]

The Supreme Court and the Federal Circuit interpreted novelty and newness as separate requirements.[53] Thus, even if an invention is new under section 101, it can still fail the novelty requirement under section 102, and vice versa.[54] On the one side, for a finding of novelty under section 102, the subject matter must not have been part of the prior art. Thus, for example, the novelty standard does not exclude from patent protection laws of nature that are not yet known. However, such laws of nature are not patent-eligible because they are not new in the sense of section 101. The underlying reasoning is that laws of nature reveal relationships that have always existed, whether or not they were previously recognized.[55] On the other side, using a law of nature in a new way (in a way that has not always existed), does not necessarily lead to a finding of novelty. For example, if such use was already known to others, then that use is not novel under section 102. Accordingly, novelty and newness must be evaluated separately.

48. 35 U.S.C. § 101 (2006).

49. *Id.* § 102.

50. *See In re* Bergy, 596 F.2d 952, 961 (C.C.P.A. 1979); *In re* Bergstrom, 427 F.2d 1394, 1401 (C.C.P.A. 1970).

51. *In re Bergy*, 596 F.2d at 961.

52. *In re Bergstrom*, 427 F.2d at 1401.

53. *See* Diamond v. Diehr, 450 U.S. 175, 211-12 (1981) (Stevens, J., dissenting) (discussing "the critical difference between the 'discovery' requirement in § 101 and the 'novelty' requirement in § 102"); Parker v. Flook, 437 U.S. 584, 593 (1978) ("The obligation to determine what type of discovery is sought to be patented must precede the determination of whether that discovery is, in fact, new or obvious."); *see also* Ambrose, *supra* note 36, at 905.

54. *In re* Nuijten, 500 F.3d 1346, 1364 (Fed. Cir. 2007) (Linn, J., dissenting) ("[A] discovery or invention can fail to be 'new' in the § 101 sense . . . even if it is 'novel' under § 102.").

55. Ambrose, *supra* note 36, at 905.

Section 101 also requires that patented subject matter be useful.[56] The term "useful" means that patent protection should be awarded in the fields of applied science, rather than in the abstract scientific fields.[57] Together with section 112, which mandates a description of the utility of the invention, section 101 represents the utility requirement of patent law.[58] According to this utility requirement, any new invention must be useful and its usefulness must be disclosed in the patent specification.[59] In other words, the invention must do something,[60] and what it does must be disclosed.[61] The invention's ability to do something is its general utility, while its particular function is its specific utility. These two utility levels are complemented by the assessment of moral utility, producing a comprehensive utility evaluation.

Based on the requirements for new and useful subject matter, the Supreme Court excluded three fundamental principles from patent eligibility: laws of nature, natural phenomena, and abstract ideas.[62] As laws of nature and natural phenomena are in existence independent of their recognition, they are not new under section 101 and are therefore ineligible for patent protection. Beyond that, abstract ideas are also nonpatentable because they are not useful. Abstract ideas fail the first level of the utility analysis because they lack general utility. Business methods[63] and mathematical algorithms[64] can possibly be excluded from patent eligibility insofar as they represent abstract ideas, laws of nature, or natural phenomena.[65]

56. 35 U.S.C. § 101 (2006).

57. Ambrose, *supra* note 36, at 907.

58. 35 U.S.C. § 112.

59. *Id.* §§ 101, 112.

60. ROBERT P. MERGES & JOHN F. DUFFY, PATENT LAW AND POLICY: CASES AND MATERIALS 211 (3d ed. 2002).

61. *See* Brenner v. Manson, 383 U.S. 519, 534-35 (1966); *In re* Fisher, 421 F.3d 1365, 1371 (Fed. Cir. 2005).

62. *E.g.*, Bilski v. Kappos, 130 S. Ct. 3218, 3225 (2010); Diamond v. Diehr, 450 U.S. 175, 185 (1981); Diamond v. Chakrabarty, 447 U.S. 303, 309 (1980).

63. *See Bilski*, 130 S. Ct. at 3228 ("Section 101 similarly precludes the broad contention that the term 'process' categorically excludes business methods."); *see also* 35 U.S.C. § 273(a)(3) ("[T]he term 'method' means a method of doing or conducting business.").

64. State Street Bank & Trust Co. v. Signature Fin. Group, Inc., 149 F.3d 1368, 1373 n.4 (Fed. Cir. 1998) ("[T]he mathematical algorithm is unpatentable only to the extent that it represents an abstract idea. . . .").

65. *See* Robert A. McFarlane & Robert G. Litts, *Business Methods and Patentable Subject Matter Following* In re Bilski*: Is "Anything Under the Sun Made by Man" Really Patentable?*, 26 SANTA CLARA COMPUTER & HIGH TECH. L.J. 35, 38 (2010).

One way to identify nonpatentable abstract ideas is to apply the printed matter doctrine.[66] According to this doctrine, text on paper or as part of any other medium generally does not constitute patent-eligible subject matter. Traditionally, the printed matter doctrine was treated as an issue of patent eligibility.[67] However, in more recent cases printed matter was addressed under the obviousness analysis of section 103.[68] The courts justified this classification on the grounds that the lack of functionality in printed matter leads to indistinguishability from the prior art.[69] Thus, the critical question for patent eligibility is whether there exists any functional relationship between the printed matter and the substrate.[70] It follows that the printed matter cases have no factual relevance where information is processed not by the mind, but by a computer.[71]

The evaluation for new and useful subject matter usually merges into the definitions for processes, machines, manufactures, and compositions of matter, creating a comprehensive analysis often referred to as a patent-eligibility test. For example, the machine-or-transformation test for the patent-eligibility of a process requires that an act or a series of acts be tied to a particular machine or transform an article into a different state or thing.[72] This test merges the analysis of newness and utility with the process definition, thereby ensuring that the claimed

66. *See generally In re* Nuijten, 500 F.3d 1346, 1365 (Fed. Cir. 2007) (Linn, J., dissenting); Dan L. Burk, *Patenting Speech*, 79 TEX. L. REV. 99, 141-45 (2000); *see also* CHISUM, *supra* note 47, § 1.024 (discussing the printed matter doctrine in the context of manufactures which was held equally applicable to processes, machines, and compositions of matter).

67. *E.g.*, *In re* Miller, 418 F.2d 1392, 1396 (C.C.P.A. 1969); *In re* Russell, 48 F.2d 668 (C.C.P.A. 1931).

68. *E.g.*, *In re* Bryan, 323 F. App'x 898, 901 (Fed. Cir. 2009); *In re* Ngai, 367 F.3d 1336, 1339 (Fed. Cir. 2004); Bloomstein v. Paramount Pictures Corp., 99-1051, 99-1203, 1999 WL 693869, at *2 (Fed. Cir. Sept. 3, 1999); *In re* Lowry, 32 F.3d 1579, 1582 (Fed. Cir. 1994); *In re* Gulack, 703 F.2d 1381, 1385 (Fed. Cir. 1983).

69. *E.g.*, *Bloomstein*, 1999 WL 693869, at *2; *In re Gulack*, 703 F.2d at 1385.

70. *See, e.g.*, *In re Bryan*, 323 F. App'x at 901; *In re Ngai*, 367 F.3d at 1339; *Bloomstein*, 1999 WL 693869, at *2; *In re Lowry*, 32 F.3d at 1582; *In re Gulack*, 703 F.2d at 1385.

71. *See In re Lowry*, 32 F.3d at 1583; *In re* Bernhart, 417 F.2d 1395, 1399 (C.C.P.A. 1969).

72. Diamond v. Diehr, 450 U.S. 175, 182-83 (1981); Parker v. Flook, 437 U.S. 584, 589 n.9 (1978); Gottschalk v. Benson, 409 U.S. 63, 71 (1972); Cochrane v. Deener, 94 U.S. 780, 788 (1877); SiRF Tech. v. Int'l Trade Comm'n, 601 F.3d 1319, 1332 (Fed. Cir. 2010); Prometheus Labs., Inc. v. Mayo Collaborative Servs., 581 F.3d 1336, 1342 (Fed. Cir. 2009); *In re* Ferguson, 558 F.3d 1359, 1363 (Fed. Cir. 2009); *In re* Bilski, 545 F.3d 943, 954 (Fed. Cir. 2008). For the first alternative of the machine-or-transformation test, the machine definition becomes relevant.

subject matter does not pre-empt the public from using the applied fundamental principles.[73]

It is difficult to distinguish exactly among the three nonpatentable fundamental principles. This is particularly true for a mathematical principle, which can be viewed as a law of nature or natural phenomenon, while the closest categorization is probably an abstract idea. However, a clear distinction among the fundamental principles is not necessary because claims for these principles are *all* excluded from patent eligibility. As long as it is sufficiently clear that claimed subject matter is comprised of nothing more than one or more fundamental principles—even if they arguably fall into multiple categories—that subject matter will be nonpatentable.

III. Application of the Law on Patent Eligibility

Having surveyed the law of patent eligibility, this Part will apply the law to programming languages and tools. However, to what subject matter do programming languages and tools actually refer? First of all, a programming language is simply a formal language. Thus, like any formal language, it is defined by its grammars.[74] The grammars determine which expressions can be used to write source code and how they can be combined for obtaining functionality. In turn, the source code written according to the grammars is recognized in programming tools for transforming such source code into executable target-machine code. The two basic tools for such transformation are compilers, which transform source code programs as a whole, and interpreters, which transform source code piece by piece during runtime.

This Part will start on familiar terrain by analyzing the patent eligibility of computer programs. The analysis will not cover computer programs in general, but will be specifically tailored to compilers, which are among the most central programming tools. In Subpart A, patent eligibility of compilers will be discussed, serving as an example for patent eligibility of all programming tools that transform source code into another code form. Subpart B will focus on patent eligibility of the language recognizers inside compilers—lexical, syntactic, and semantic analyzers. This analysis is followed in Subpart C by an evaluation of patent eligibility of the language recognizers' corresponding grammars,

73. *See, e.g.*, Bilski v. Kappos, 130 S. Ct. 3218, 3258 (2010); Funk Bros. Seed Co. v. Kalo Inoculant Co., 333 U.S. 127, 130 (1948); *In re* Nuijten, 500 F.3d 1346, 1364 (Fed. Cir. 2007) (Linn, J., dissenting); CHISUM, *supra* note 47, § 1.03[b][k][VII].

74. In the following, "grammar" is used as a generic term encompassing lexical (regular), context-free, and attribute grammars. *See infra* Part III.C.

which, if deemed patent-eligible, would also lead to patents on programming languages. This Part will then conclude with a patent eligibility analysis of additional programming tools in Subpart D.

A. Compilers

A computer program written in a programming language, for instance in source code format, is not immediately executable; rather, it must be first converted into target machine code.[75] The compiler responsible for this task satisfies the machine definition. Applying *Alappat*, a compiler turns a general purpose computer into a particular machine for converting source code to target machine code.[76] In the words of Professor Samuelson, a compiler *is* a machine for converting source code into target machine code.[77] Thus, a compiler is patent-eligible under the same requirements as any other computer program. It is a computer program with the same characteristics as other computer programs, such as word processing or spreadsheet programs. It is a machine for creating other machines. Thus, the patent eligibility of programming tools can be reduced to the general question of patent eligibility of computer programs.

A compiler also satisfies the machine-or-transformation test, making it patent-eligible as a process. According to the machine-or-transformation test, an act or a series of acts is patent-eligible as a process if it is tied to a particular machine or apparatus, or transforms a particular article into a different state or thing. First, the requirement for an act or a series of acts is satisfied because a compiler is a computer program consisting of an algorithm. An algorithm is defined as an unambiguous specification of a conditional sequence of steps or operations for solving a class of problems.[78] Thus, by definition, a compiler consists of a series of acts. Consequently, the first requirement for a process—presence of an act or a series of acts—is satisfied.

Further, the machine prong of the machine-or-transformation test is satisfied. While an unspecified reference to a general-purpose computer

75. Target machine code can be object code to be executed on a particular type of physical computer. However, target machine code can also be an intermediate code for a virtual machine, which then in turn is executed on a physical computer.

76. *In re* Alappat, 33 F.3d 1529, 1545 (Fed. Cir. 1994).

77. *See* Samuelson et al., *supra* note 21, at 2320.

78. *In re* Schrader, 22 F.3d 290, 293 n.5 (Fed. Cir. 1994) (citing Allen Newell, *The Future of Software Protection: Response: The Models Are Broken, the Models Are Broken*, 47 U. PITT. L. REV. 1023, 1024 (1986)).

does not tie an act or a series of acts to a particular machine,[79] it is possible to create such a tie by a specific programming of the general purpose computer.[80] In the case of a compiler program, its programming creates a particular machine for converting source code into target machine code, thereby tying the compiler program to a particular machine. Arguably, under Professor Samuelson's view of programs as machines, the compiler is not tied to a particular machine, but rather *is* a particular machine; however, recognizing programs as machines themselves does not negate the possibility that they still can be tied to other machines, such as the general purpose computers on which they are executed.

According to the transformation prong of the machine-or-transformation test, transformation and reduction of an article to a different state or thing is the "clue" to patent eligibility.[81] Under this requirement, a compiler is patent-eligible as a process because it takes a general purpose computer and creates a special purpose computer for transforming source code into target machine code. Also, a compiler can turn other programs into a different state or thing. For example, as part of an Integrated Development Environment (IDE) for multiple programming languages,[82] adding a compiler for a particular language that is not yet supported will create a different machine because the IDE will be able to compile a new language. Thus, the transformation prong of the machine-or-transformation test is satisfied.

Not only does the machine-or-transformation test support the finding that compilers are patent-eligible, but other tests lead to the same result as well. Under the contemporary view of the mental steps doctrine, compilers are not excluded from patent eligibility because their execution does not depend on any human decision making. The Freeman-Walter-Abele, physical steps, and "useful, concrete and tangible result" tests would also be satisfied. The common requirement for some physical or tangible characteristic would be fulfilled because a compiler changes the memory of a computer. Specifically, the parser within the

---

79. Accenture Global Servs. GmbH v. Guidewire Software, Inc., 691 F. Supp. 2d 577, 597 (D. Del. 2010); DealerTrack, Inc. v. Huber, 657 F. Supp. 2d 1152, 1155-56 (C.D. Cal. 2009); Fuzzysharp Techs. Inc. v. 3D Labs, Inc., C07-5948 SBA, 2009 U.S. Dist. LEXIS 115493, at *12-13 (N.D. Cal. Dec. 11, 2009).

80. *Cf. DealerTrack*, 657 F. Supp. 2d at 1156 ("The [patent] does not specify how the computer hardware and database are 'specially programmed' to perform the steps claimed in the patent . . . and the claimed central processor is nothing more than a general purpose computer that has been programmed in some unspecified manner.").

81. Bilksi v. Kappos, 130 S. Ct. 3218, 3227 (2010); Diamond v. Diehr, 450 U.S. 175, 184 (1981); Gottschalk v. Benson, 409 U.S. 63, 70 (1972).

82. *See infra* Part III.D.

compiler generates a parse tree or other data structure for arranging memory space.[83] Also, under the technological arts test, compilers would qualify as patent-eligible processes because software design and engineering is a computer science discipline and not exclusively a domain of the liberal arts.

A compiler satisfies the definition of a manufacture as well. The required transformation of raw materials consists of a conversion of characters of the alphabet into a source code text that can be processed by a compiler program. To create an executable compiler program, the source code of the compiler has to be compiled itself. In this regard, the arrangement of the characters in the source code creates a new instrument bearing the name "Compiler." The new instrument exhibits functionality that was not present before the arrangement and combination of the characters. It is functional because it can be compiled into machine code, that is, code executable on a particular computer. Then the compiled code can be actually executed in a computer. Consequently, writing a compiler creates an article of manufacture.

A final obstacle for patent eligibility of compilers could be the printed matter doctrine. Because a compiler in source code format consists of text, the printed matter doctrine arguably supports the conclusion that the compiler is merely a nonpatentable abstract idea. However, if text is to be processed not by the mind, but by a machine, the printed matter doctrine is satisfied because the machine is using the printed matter for performing a function. The text and the medium work in tandem to bring about a particular result. Specifically, in the case of a compiler, the text is a building block for creating a language converter. As such, it not only represents information, but exhibits functionality. Thus, the printed matter doctrine is satisfied, leading to the conclusion that compilers are patent-eligible as machines, processes, and manufactures.

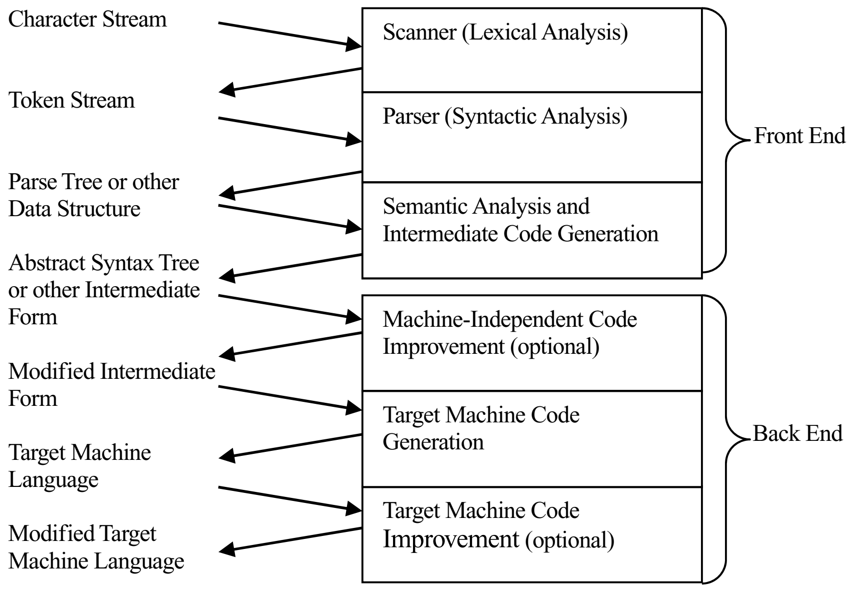
B. Programming Language Recognizers

Compilers transform source code in multiple phases. First, a scanner performs a lexical analysis to ensure that the code makes proper use of the lexical grammar, or the vocabulary of the language.[84] The scanner partitions the code into tokens, which represent the shortest character strings with individual meaning.[85] Thereafter, in the syntactic

83. *See infra* Part III.B.
84. *See infra* Part III.C. A "scanner" is also commonly referred to as "lexer."
85. MICHAEL L. SCOTT, PROGRAMMING LANGUAGE PRAGMATICS 43 (3d ed. 2009).

analysis, a parser creates a parse tree or other data structure from the tokens. In the following semantic analysis, the data is given meaning by adding attributes. After the semantic analysis, all phases for determining the meaning of the source code are carried out. Now the intermediate code form is passed from the compiler front end to the back end for generating the target machine code. The separation of front and back end is useful for creating a multilanguage compiler family. Because the intermediate code form is independent of both programming language and target machine language, it is possible to build n front ends and m back ends instead of n×m integrated compilers.

Organization of a Typical Compiler
(modified from MICHAEL L. SCOTT, PROGRAMMING LANGUAGE PRAGMATICS 26 (3d ed. 2009))


Character Stream
Token Stream
Parse Tree or other Data Structure
Abstract Syntax Tree or other Intermediate Form
Modified Intermediate Form
Target Machine Language
Modified Target Machine Language
Scanner (Lexical Analysis)
Parser (Syntactic Analysis)
Semantic Analysis and Intermediate Code Generation
Machine-Independent Code Improvement (optional)
Target Machine Code Generation
Target Machine Code Improvement (optional)
Front End
Back End

The single phases of a compiler front end are patent-eligible as language recognizers. Each phase can be categorized as a machine under section 101. For example, a scanner is a machine that recognizes whether source code written in a particular programming language conforms to the lexical grammar for that programming language.[86] If a particular source code fragment is recognized, the scanner will pass a

86. *See* Larry Morell & David Middleton, *Recursive-Ascent Parsing*, 18 J. COMPUTING SCI. C. 186, 187 (2003).

corresponding token to the parser. If the source code fragment is not recognized, it will output an error message. In this respect, a scanner is a machine for conducting a lexical analysis. It is a language recognizer and is patent-eligible as such. Similarly, a parser is patent-eligible as a machine for recognizing the syntactic structure of a source code program and a semantic analyzer as a machine for recognizing the semantics of a program. Scanners, parsers, and semantic analyzers are also patent-eligible as processes and manufactures for the equivalent reasons that compilers are patent-eligible as processes and manufactures.[87]

C. Grammars and Programming Languages

Scanners, parsers, and semantic analyzers recognize source code written according to their corresponding grammars. More precisely, scanners correspond to lexical grammars, parsers correspond to context-free grammars, and semantic analyzers correspond to attribute grammars. These three grammars comprehensively specify the rules for building valid expressions in a particular programming language. In other words, they define and represent the programming language. Thus, if lexical, context-free, and attribute grammars are considered patent-eligible subject matter, the result would be the patent eligibility of programming languages. The patent eligibility of grammars would lead in its very essence to what truly could be called a "patent on a programming language."

While scanners, parsers, and semantic analyzers work as language *recognizers*, their corresponding grammars can be qualified as language *generators*.[88] The relationship between recognizers and generators can be illustrated by the language of a calculator. In the lexical grammar below, the tokens on the left side of the arrow generate the characters on the right side. For example, the token "minus" generates the "-" sign and the token "lparen" generates a left parenthesis. Further, the token "number" generates all decimal digits with at most one decimal point. The rule states the following: "number" ("number") generates ("→") a digit ("digit") followed by zero or more digits ("digit*") or alternatively generates ("|") zero or more digits ("digit*"), followed by a decimal point and a digit ("(. digit)"), followed by zero or more digits ("digit*").

87. *See supra* Part III.B.
88. Scott, *supra* note 85, at 101.

Lexical Grammar of a Calculator Language
(modified from Michael L. Scott, Programming Language Pragmatics ch. 2.2., at 52 (3rd ed. 2009))

```
assign → :=
plus → +
minus → -
times → *
div → /
lparen → (
rparen →)
number → digit digit* | digit* ( . digit) digit*
```

Having discussed a generator for the calculator language, its corresponding recognizer can be implemented into software by means of the scanner given in pseudo code below. The scanner checks the validity of each single input through a series of if-statements. Whenever an input satisfies a condition of an if-statement, the body of the if-statement is executed and a token is returned to the parser. For example, if the scanner reads a digit, it will continue reading further digits and at most one decimal point. If no further decimal point is read, the token "number" is returned to the parser. However, if the input is invalid under the calculator language, for instance, if it reads "0.1.1," the scanner will not accept the input, but instead will announce an error. After evaluating an input, the scanner algorithm will be repeated from the beginning by evaluating the next input until all inputs are read.[89]

Pseudo code Implementation of a Scanner for the Calculator Language
(modified from Michael L. Scott, Programming Language Pragmatics ch. 2.2., at 53 (3rd ed. 2009))

```
skip any initial white space (spaces, tabs, and newlines)
if cur_char Element of {'(' ')' '+' '-' '*' '/'}
    return the corresponding token
if cur_char = ':'
    read the next character
    if it is '='
        then return assign
    else announce an error
if cur_char = '.'
    read the next character
    if it is a digit
        read any additional digits
        return number
    else announce an error
```

89. Scott, *supra* note 85, at 53.

```
if cur_char is a digit
    read any additional digits and at most one decimal point
    return number
else announce an error
```

The pseudo code demonstrates that the calculator scanner corresponds to the lexical grammar of the calculator language. Generally, in order for a scanner to properly identify whether a certain input belongs to a particular programming language, it must be designed according to the lexical grammar of that language. In this sense, the scanner is the mirror image of the grammar. Such correlation does not only exist between scanners and lexical grammars, but also between parsers and context-free grammars, and between semantic analyzers and attribute grammars. These correlations are the basis for automated compiler development by means of computer programs, such as compiler compilers or parser generators. For example, a parser generator takes as input a context-free grammar and returns the source code of a parser for that grammar.

Although scanners and other language recognizers are patent-eligible, this finding does not necessarily imply that grammars are patent-eligible as well. Rather, the opposite is true. First of all, grammars do not match the traditional machine definition, which is limited to mechanical devices or other industrial equipment. Grammars also do not qualify as machines under *Alappat* because they are not implemented in computer programs. Instead, grammars are typically contained in a programming language specification document that can be used by software designers for writing computer programs in conformity with the grammar. If their source code complies with the grammar, a respective compiler will accept and convert the source code into target machine code. Therefore, there is no software implementation of grammars. For this reason, grammars are also excluded from patent eligibility under Professor Samuelson's view of computer programs as machines.

Further, it is doubtful whether grammars can be interpreted as processes. For one thing, it is inadequate to categorize a grammar as an act or a series of acts. For example, the generation of a parse tree for a given program consists of a number of steps. However, grammars are better categorized as sets of rules rather than as ordered series of steps.[90]

90. *But see* Hamilton & Sabety, *supra* note 2, at 263 n.95 (arguing that a computer language can be considered an algorithmic process for converting one set of symbols (source code) into another set (object code)).

Grammars do not imply that certain things be done with certain substances in a certain order, which would be necessary to deem them an act or a series of acts. Returning to the example of the lexical grammar for the calculator language, the only order involved is the reading of the grammar from left to right. The tokens on the left can generate the characters on the right. This simple order of precedence appears to be insufficient for qualifying the grammar as an act or series of acts as required by the process definition.

Even assuming that a grammar can be categorized as an act or a series of acts, it would still fail both the machine and transformation prong of the machine-or-transformation test. First, the machine prong is not satisfied because a grammar is not tied to a particular machine, nor is it implemented in a computer program. Second, concerning the transformation prong, the grammar's application, rather than the grammar itself, performs the transformation. For instance, in the case of the calculator language, the lexical grammar merely represents abstract rules. It states which tokens can generate which characters. However, this is not a transformation in itself, but rather a simple statement of which transformations are possible.

Assuming—contrary to the foregoing—that grammars satisfy the process definition, the mental steps doctrine would become relevant. In this regard, the Federal Circuit's current interpretation of the mental steps doctrine would not exclude grammars from patent eligibility. After all, grammars are not dependent on human thinking or decision making. Rather, the opposite holds true. Grammars can be thought of as sets of clear and definite rules that leave no ambiguity to be resolved by human thinking and decision making. For example, in the case of the lexical grammar for the calculator language, the token "plus" generates the "+" sign. This rule for generating a "+" sign is unequivocal, leaving no discretion for human thinking in the sense of the mental steps doctrine.

A grammar cannot be claimed as a manufacture because the manufacture definition covers only tangible subject matter. Accordingly, it must be connected to a medium, such as a programming language specification document. This requirement, in turn, leads to application of the printed matter doctrine, which mandates a functional relationship between the printed matter and the medium. However, in whatever medium the grammar is manifested, it is hardly imaginable that it exhibits a functional relationship to the medium. Particularly, the grammar's lack of software implementation impedes functionality and hence, patent eligibility. On a side note, apart from the technological arts test, the printed matter doctrine would apply in the same way to the earlier patent eligibility tests due to their respective physicality and tangibility requirements.

Having reached the end of the analysis for grammars, it can be concluded that under the current law of patent eligibility, grammars are not patent-eligible subject matter. This conclusion can be drawn because all tests for patent eligibility are in some way limited to tangible or physical subject matter. The doctrinal ground for these limitations is founded in the attempt to avoid patenting natural phenomena, laws of nature, and abstract ideas.[91] Grammars represent abstract ideas, the embodiment of which, in whatever tangible medium, is prevented from patent eligibility by the printed matter doctrine. As grammars are the defining elements of programming languages, it follows that programming languages are not patent-eligible.

Programming languages are patent-ineligible. However, from a patent law perspective, the practical relevance of this finding appears to be low. Because source code written according to a particular programming language's grammars can be only recognized in corresponding compilers or other language recognizers, obtaining a patent on such recognizers gives the patent holder the same control over the software implementation of the programming language as if its grammars were patent-eligible. In the case of a patent on recognizers for that programming language, third parties would be effectively prohibited from writing any language-specific programming tools. Thus, the patent eligibility of compilers and other recognizers has the same effect that the patent eligibility of programming languages would have.

D. Further Programming Tools

Typically, programming languages are supported by a variety of programming tools beyond compilers or other language transformers. Dissemination and use of a programming language is often critically dependent on the available tools. In this regard, target machine developers usually provide a software development kit (SDK) in order to support the writing of programs that run on their target machine.[92] An SDK can contain or interact with an IDE, which is a comprehensive programming tool for developing programs in a particular programming language. Usually, an IDE combines a text editor for writing and editing source code, a debugger for correcting syntactic errors in the source

91. *See* Gibby, *supra* note 21, at 341.
92. SDKs are also commonly referred to as "software development environments (SDEs)."

code, a compiler for transforming the source code into machine-readable code, and a linker for connecting source code to elements in a programming library.

Patent eligibility of a programming tool can refer to complex software systems, such as SDKs or IDEs, but also to smaller programs, such as parsers or even data structures. For purposes of patent eligibility it is irrelevant whether code is executable as a standalone program.[93] Rather, the decisive inquiry is whether executable code can be categorized as a new and useful machine, process, or manufacture. In line with this rule but deviating from its earlier decisions,[94] the Federal Circuit held in *In re Lowry* that data structures are patent-eligible as manufactures, stressing their characteristic physical property of providing increased efficiency in computer operations by physically optimizing memory.[95]

Programming tools can be patent-eligible as machines, processes, and manufactures under the same standards as discussed for compilers.[96] Thus, for example, SDKs can be patent-eligible as machines for facilitating software development for a particular target machine. Similarly, data structures can be patent-eligible as machines for logically and physically organizing storage of data in memory.[97] SDKs and data structures are further patent-eligible as processes and manufactures for the equivalent reasons that compilers are patent-eligible as such.[98] These findings are equally true for the programming tools listed in the following table, which, however, is not intended to be a comprehensive reflection of patent-eligible programming tools.

93. This is equally true for copyright eligibility, as indicated by the statutory definition of computer programs. *See* 17 U.S.C. § 101 (2006).

94. *See In re* Warmerdam, 33 F.3d 1354, 1361-62 (Fed. Cir. 1994); *In re* Trovato, 42 F.3d 1376 (Fed. Cir. 1994), *vacated*, 60 F.3d 807 (Fed. Cir. 1995).

95. *In re* Lowry, 32 F.3d 1579, 1583-84 (Fed. Cir. 1994); *see also* Carl Chan, Note, *The Patentability of Software Data Structures After* Lowry *and* Warmerdam, 32 NEW ENG. L. REV. 899 (1998); Hamilton & Sabety, *supra* note 2, at 254-59; Andrew J. Hollander, *Patenting Computer Data Structures: The Ghost, the Machine and the Federal Circuit*, 2003 DUKE L. & TECH. REV. 33; Steven M. Santisi, Note, In re Warmerdam*: When Is a Software Process Too Abstract To Merit Patent Protection?*, 13 J. MARSHALL J. COMPUTER & INFO. L. 667 (1995); Brian R. Yoshida, *Claiming Electronic and Software Technologies: The Effect of the Federal Circuit Decisions in* Alappat*,* Warmerdam*, and* Lowry *on the Claiming of Mathematical Algorithms and Data Structures*, 45 BUFF. L. REV. 457 (1997).

96. *See supra* Part III.A.

97. *See generally* Hollander, *supra* note 95, ¶ 9-16.

98. *See supra* Part III.A.

PROGRAMMING TOOL	MACHINE FOR. . . / PROCESS TIED TO A GENERAL PURPOSE COMPUTER TURNING IT INTO A MACHINE FOR . . .
Application Programming Interface (API)	connecting a computer program to a programming library.
Code Optimizer	making a computer program run more efficiently.
Compiler	transforming source code into machine-readable code before execution of the source code.
Compiler Compiler	generating a compiler or parts thereof from a formal specification, such as a grammar.
Data Structure	logically and physically organizing storage of data in memory in a certain way.
Data Type	determining the range of values for a piece of data.
Debugger	finding bugs in source code.
IDE	facilitating software development in a particular programming language.
Interpreter	transforming source code into machine-readable code during execution of the source code.
Linker	connecting elements generated by a compiler with code contained in programming libraries forming an executable program.
Parser	recognizing the syntactic structure of a source code program.
Programming Library	implementing auxiliary code to be used in connection with programs written in the language the library was created for.
Scanner	recognizing the lexical structure of a source code program.
SDK / SDE	facilitating software development for a particular target machine.
Semantic Analyzer	recognizing the semantic structure of a source code program.
Target Code Generator	transforming code from intermediate format into machine-readable code.
Text Editor	writing source code.
Translator	transforming source code of one programming language into source code of another programming language.
Type System	categorizing data into predefined data types.

IV. FREE SPEECH LIMITATION OF PATENT ELIGIBILITY

According to section 271(a), "whoever without authority makes, uses, offers to sell, or sells any patented invention, within the United States, or imports into the United States any patented invention during the term of the patent therefor, infringes the patent."[99] For example, in the case of a patent on a compiler for a particular programming language, the patent holder could prohibit any third party from making such a compiler. Because the making of a compiler involves programming, in other words, involves writing a text, such a patent could lead to an

99. 35 U.S.C. § 27(a) (2006).

unconstitutional restriction of speech.[100] If that were the case, a different interpretation of patent-eligible subject matter would be necessary because if an otherwise acceptable construction of a statute would raise serious constitutional problems, the statute must be construed to avoid such problems unless such construction is plainly contrary to the intent of Congress.[101]

Similar to a patent on any ordinary computer program, a patent on a programming tool creates a potentially unconstitutional free speech restriction. For example, in the case of a patent on a compiler for a particular programming language, the patent holder could exclude third parties from writing other compilers to the extent the compiler is claimed in the patent. However, because of the ineligibility of programming languages under section 101, there is no patent right to exclude third parties completely from using a particular programming language. Thus, holding programming languages patent-ineligible already substantially reduces the potential chilling effects of section 101 on free speech.

Against this background, this Part will discuss whether the First Amendment requires an interpretation of section 101 that excludes programming tools. The first Subpart A will briefly discuss whether the First Amendment is applicable. Subpart B will evaluate whether source code qualifies as speech. Subpart C will discuss whether a restriction based on section 101, such as a prohibition to write a particular program, would be a content-based or content-neutral speech restriction, if any restriction at all. Subpart D will analyze whether a patent on a programming tool could lead to a prior restraint, restricting speech before it actually occurs. Finally, Subpart E will address whether section 101 is a vague and overbroad law.

A. Applicability of the First Amendment

Because the restriction of speech in the case of a patent on a programming tool would typically be based on the right of a private party, it is doubtful that the First Amendment is applicable to programming tools. Generally, First Amendment rights can only be asserted against the government.[102] However, the state action doctrine can

100. Burk, *supra* note 66, at 150; Thomas F. Cotter, *A Burkean Perspective on Patent Eligibility*, 22 BERKELEY TECH. L.J. 855, 880 (2007).

101. *See, e.g.*, Solid Waste Agency of N. Cook Cnty. v. U.S. Army Corps of Eng'rs, 531 U.S. 159, 173 (2001) (quoting Edward J. DeBartolo Corp. v. Fla. Gulf Coast Bldg. & Constr. Trades Council, 485 U.S. 568, 575 (1988)).

102. *See, e.g.*, Lloyd Corp. v. Tanner, 407 U.S. 551, 567 (1972); *see also* Andrew F. Knight, *A Patently Novel Plot: Fiction, Information, and Patents in the Twenty-First Century*, 47 IDEA

render the First Amendment applicable. Under the state action doctrine, the conduct of private parties can interfere with constitutional guarantees if it is sufficiently entwined with a governmental entity.[103] Consequently, in the case of enforcement of a patent on a programming tool, the First Amendment would apply if such enforcement is entwined with a governmental entity. This view harmonizes copyright and patent law as it is prevailing opinion that the enforcement of copyrights by private parties can constitute a free speech restriction.[104]

B. Source Code as Speech

The First Amendment to the United States Constitution states that "Congress shall make no law . . . abridging the freedom of speech."[105] Thus, in order to fall within scope of the First Amendment, source code must be considered speech. Indeed, most courts have categorized expressions in programming languages as speech or recognize that it might warrant First Amendment protection.[106] In this regard, it can be argued that communication does not lose constitutional protection as speech simply because it is expressed in the form of programming language.[107] Based on this reasoning, some commentators view code as speech as well.[108] Others have expressed their disagreement,[109] which

203, 223 (2006); John R. Thomas, *The Future of Patent Law: Liberty and Property in the Patent Law*, 39 HOUS. L. REV. 569, 592 (2002).

103. *See, e.g.*, Shelley v. Kraemer, 334 U.S. 1, 20 (1948); *see also* Thomas, *supra* note 102, at 592.

104. *See e.g.*, Zacchini v. Scripps-Howard Broad. Co., 433 U.S. 562, 575-76 (1977); Julie E. Cohen, *A Right To Read Anonymously: A Closer Look at "Copyright Management" in Cyberspace*, 28 CONN. L. REV. 981, 1021 (1996); Wendy J. Gordon, *A Property Right in Self-Expression: Equality and Individualism in the Natural Law of Intellectual Property*, 102 YALE L.J. 1533 1607 n.400 (1993); Mark A. Lemley & Eugene Volokh, *Freedom of Speech and Injunctions in Intellectual Property Cases*, 48 DUKE L.J. 147, 182, 206 (1998); Mel Marquis, Comment, *Fair Use of the First Amendment: Parody and Its Protections*, 8 SETON HALL CONST. L.J. 123, 136 (1997). *See generally* Jennifer E. Rothman, *Liberating Copyright: Thinking Beyond Free Speech*, 95 CORNELL L. REV. 463, 508-09 (2010).

105. U.S. CONST. amend. I.

106. *See* Universal City Studios, Inc. v. Corley, 273 F.3d 429, 445-46 (2d Cir. 2001); Junger v. Daley, 209 F.3d 481, 485 (6th Cir. 2000); Bernstein v. U.S. Dep't of Justice, 176 F.3d 1132, 1142 (9th Cir. 1999); United States v. Mendelsohn, 896 F.2d 1183, 1186 (9th Cir. 1990); Universal City Studios v. Reimerdes, 111 F. Supp. 2d 294, 326-27 (S.D.N.Y. 2000); Bernstein v. U.S. Dep't of State, 945 F. Supp. 1279, 1287 (N.D. Cal. 1996); Bernstein v. U.S. Dep't of State, 922 F. Supp. 1426, 1436 (N.D. Cal. 1996); DVD Copy Control Ass'n, Inc. v. Bunner, 75 P.3d 1, 10 (Cal. 2003).

107. *Corley*, 273 F.3d at 445.

108. Norman A. Crain, Comment, Bernstein, Karn, *and* Junger*: Constitutional Challenges to Cryptographic Regulations*, 50 ALA. L. REV. 869, 887-88 (1999); Fitzgerald, *supra* note 5, at 352; Steven E. Halpern, *Harmonizing the Convergence of Medium, Expression, and*

could be justified by emphasizing that writing source code is equivalent to constructing a machine.[110]

The difficulty of determining whether source code can be qualified as speech is rooted in its dual character as both a functional machine and means of expression. Writing source code unites functionality and expression, making a program eligible for patent protection on the one hand while subjecting it to First Amendment scrutiny on the other. However, this duality of functionality and expression is not unique to writing computer programs. To some extent, expression and action are always merged.[111] Consequently, some kernel of expression can be found in almost every activity a person undertakes, but such a kernel is not sufficient to bring the activity within the protection of the First Amendment.[112] Rather, the question becomes where to draw the line for finding that an object resulting from a particular conduct or an activity is sufficiently expressive to warrant protection by the First Amendment.

The Supreme Court's *Spence* test answers this question by determining whether conduct is sufficiently imbued with elements of communication.[113] In *Spence v. Washington*, the Court laid out three determinative factors: (1) the nature of the activity, (2) its factual context, and (3) the environment in which it was undertaken.[114] The *Spence* test evolved into the message test, which consists of determining whether symbolic acts are intended to communicate a particularized message and deciding if there is a great likelihood that the message would be understood.[115] The *Spence* test and the message test work

---

Functionality: A Study of the Speech Interest in Computer Software, 14 HARV. J. L. & TECH. 139, 150, 158 (2000).

109. John P. Collins, Note, *Speaking in Code*, 106 YALE L.J. 2691, 2696 (1997); Seth Hanson, Note, Bernstein v. United States Dep't of Justice*: A Cryptic Interpretation of Speech*, 2000 B.Y.U. L. REV. 663, 664; Katherine A. Moerke, Note, *Free Speech to a Machine? Encryption Software Source Code Is Not Constitutionally Protected "Speech" Under the First Amendment*, 84 MINN. L. REV. 1007, 1043 (2000).

110. Moerke, *supra* note 109, at 1045.

111. R. Polk Wagner, Note, *The Medium Is the Mistake: The Law of Software for the First Amendment*, 51 STAN. L. REV. 387, 394 (1999).

112. City of Dallas v. Stanglin, 490 U.S. 19, 25 (1989).

113. Spence v. Washington, 418 U.S. 405, 409 (1974).

114. *Id.* at 409-10; *see also* James M. McGoldrick, Jr., *Symbolic Speech: A Message from Mind to Mind*, 61 OKLA. L. REV. 1, 34-35 (2008).

115. *See, e.g.*, Texas v. Johnson, 491 U.S. 397, 404 (1989) (quoting *Spence*, 418 U.S. at 410-11); Clark v. Cmty. for Creative Non-Violence, 468 U.S. 288, 305 (1984) (quoting *Spence*, 418 U.S. at 410-11). *But see, e.g.*, Hurley v. Irish-Am. Gay, Lesbian & Bisexual Group of Bos., 515 U.S. 557, 569 (1995) ("[A] narrow, succinctly articulable message is not a condition of constitutional protection.").

independently and as a unit;[116] the message test can be used to evaluate the first factor of the *Spence* test.

When applying either test, it becomes clear that it cannot strictly be decided whether source code is speech. Rather, source code can be speech in some circumstances, and not speech in other circumstances.[117] For example, if source code is simply compiled and executed on a target machine, it is merely used functionally. However, if it is passed among programmers of a software development team to add further code, for instance, or if it is published in an academic paper, the use of the code can certainly inhibit elements of speech. Without addressing further details, it is sufficient for purposes of this Article to maintain that source code cannot be categorically excluded from being considered speech.

C. Content-Based and Content-Neutral Speech Restrictions

Having established that source code can be protected speech, it does not necessarily follow that granting a patent on a programming tool under section 101 is unconstitutional.[118] Rather, constitutionality depends on the scope of First Amendment protection: Content-based restrictions must meet strict scrutiny,[119] while content-neutral restrictions only need to pass intermediate scrutiny.[120] Professors Mark A. Lemley and Eugene Volokh argue that injunctions based on intellectual property rights are content-based restrictions[121] because they prohibit speech on the basis of a particular expression.[122] However, this argument was rebutted by

116. McGoldrick, *supra* note 114, at 79.

117. *See* Lee Tien, *Publishing Software as a Speech Act*, 15 BERKELEY TECH L.J. 629, 638-39 (2000); Ryan Christopher Fox, Comment, *Old Law and New Technology: The Problem of Computer Code and the First Amendment*, 49 UCLA L. REV. 871, 915 (2002); Robert Plotkin, *Fighting Keywords: Translating the First Amendment To Protect Software Speech*, 2003 U. ILL. J.L. TECH. & POL'Y 329, 341 (2003); Robert Post, *Encryption Source Code and the First Amendment*, 15 BERKELEY TECH. L.J. 713, 716, 720 (2000). But some courts have found source code generally to be protected speech. *See, e.g.*, Junger v. Daley, 209 F.3d 481, 485 (6th Cir. 2000); *see also* Lemley & Volokh, *supra* note 104, at 150 n.5.

118. *See* Crain, *supra* note 108, at 889.

119. *See, e.g.*, Boos v. Barry, 485 U.S. 312, 321-22 (1988) ("[W]e have required the State to show that the 'regulation is necessary to serve a compelling state interest and that it is narrowly drawn to achieve that end.'" (quoting Perry Educ. Ass'n v. Perry Local Educ. Ass'n, 460 U.S. 37, 45 (1983)).

120. *See, e.g.*, Turner Broad. Sys., Inc. v. Fed. Commc'ns Comm'n, 520 U.S. 180, 189 (1997) ("A content-neutral regulation will be sustained under the First Amendment if it advances important governmental interests unrelated to the suppression of free speech and does not burden substantially more speech than necessary to further those interests.").

121. *See* Lemley & Volokh, *supra* note 104, at 186-87; *see also* Eugene Volokh, *Freedom of Speech and Intellectual Property: Some Thoughts After* Eldred*,* 44 Liquormart*, and* Bartnicki, 40 HOUS. L. REV. 697, 710 (2003).

122. Volokh, *supra* note 121, at 703.

Professors Neil W. Netanel and Erwin Chemerinsky, stating that enforcement of a content-sensitive law does not automatically lead to a content-based restriction.[123] This is illustrated by copyright law, which applies to all speech whatever its topic or ideology.[124] Consequently, section 101 is a content-neutral law as well.

However, Professor Volokh argues that even if courts categorize intellectual property laws as content-neutral restrictions, those laws still cannot be defended under the more lenient intermediate scrutiny standard applicable to content-neutral time, place, or manner speech restrictions.[125] He claims that intellectual property laws go beyond the scope of these restrictions because they do not leave speakers adequate alternative channels to convey their expression.[126] In this regard, the standard for time, place, or manner restrictions requires that those restrictions serve a significant governmental interest, do not make reference to the content of the regulated speech, and are narrowly tailored, leaving open ample alternative channels for communication.[127] This intermediate scrutiny standard is also applicable to incidental content-neutral restrictions, like section 101, which are not aimed at restricting the time, place, or manner of speech.[128]

Despite Professor Volokh's claim, the intermediate scrutiny standard can be satisfied by section 101. First, section 101 serves a significant governmental interest, which is the constitutionally legitimated goal "[t]o promote the Progress of Science and useful Arts, by securing for limited Times to Authors and Inventors the exclusive Right to their respective Writings and Discoveries." Further, section 101 does not reference the content of speech, but rather restricts speech as an incidental consequence of granting exclusive rights in the functionality of computer programs. However, the requirement that patent protection be narrowly tailored, leaving open ample alternative channels for communication, is more difficult to satisfy. This is especially true because patent law does

123. *See* Neil Weinstock Netanel, *Locating Copyright Within the First Amendment Skein*, 54 STAN. L. REV. 1, 48 (2001); *see also* Erwin Chemerinsky, *Balancing Copyright Protections and Freedom of Speech: Why the Copyright Extension Act Is Unconstitutional*, 36 Loy. L.A. L. REV. 83, 93 (2002).

124. Chemerinsky, *supra* note 123, at 93.

125. Volokh, *supra* note 121, at 711.

126. *Id.* at 712.

127. *See, e.g.*, Ward v. Rock Against Racism, 491 U.S. 781, 791 (1989).

128. *See, e.g.*, United States v. O'Brien, 391 U.S. 367, 377 (1968) ("[A] government regulation is sufficiently justified if it is within the constitutional power of the Government; if it furthers an important or substantial governmental interest; if the governmental interest is unrelated to the suppression of free expression; and if the incidental restriction on alleged First Amendment freedoms is no greater than is essential to the furtherance of that interest.").

not employ free speech exceptions such as the fair use doctrine in copyright law.[129]

Patent law, unlike copyright law, was not designed to accommodate First Amendment interests because it traditionally granted rights on physical objects or processes unrelated to any speech.[130] However, the embrace of computer programs and other information technology has led to an increasing number of patents incidentally related to speech. The patent holder's exclusion of third parties from using a patented programming tool is one example of patent law's extension into the realm of copyright protection. Therefore, to protect First Amendment rights, fair use could be extended to patent law.[131] Also, free speech could be protected by patent misuse, equitable estoppel, an implied license based on equitable estoppel,[132] or a standards estoppel doctrine based on traditional common law principles.[133] If these principles are applied as necessary, section 101 will pass intermediate scrutiny.

D. Patents on Programming Tools as Prior Restraints

Arguably, interpreting section 101 to cover programming tools could lead to a violation of the First Amendment as a prior restraint.[134] A prior restraint is a speech restriction that becomes effective before the speech actually occurs. It can be the result of a governmental licensing scheme that gives the government the right to demand a license or permit speech to occur.[135] It can also follow from an administrative system that allows for judicial orders or injunctions.[136] On this basis, Professor Marci A. Hamilton and Ted Sabety argue that by authorizing protection for programming languages, the Copyright Act would authorize prior restraints.[137] This argument is based on the notion that a copyright is a property right allowing its holder to exclude third parties, thereby

129. *See* 17 U.S.C. § 107 (2006).

130. *See* Dan L. Burk, *Constitutional Issues Involving Use of the Internet: Software as Speech*, 8 SETON HALL CONST. L.J. 683, 691 (1998); Lemley & Volokh, *supra* note 104, at 234.

131. Burk, *supra* note 66, at 150.

132. *See* Wendy Milanese, Comment, *The Tension Must Break: The Irreconcilable Interplay Between Antitrust, Defenses to Infringement and Protection of Standardized Software Development Tools*, 15 SANTA CLARA COMPUTER & HIGH TECH. L.J. 407, 425 (1999).

133. *See* Robert P. Merges & Jeffrey M. Kuhn, *An Estoppel Doctrine for Patented Standards*, 97 CAL. L. REV. 1 (2009).

134. *See* Hamilton & Sabety, *supra* note 2, at 270.

135. *See* Near v. Minnesota, 283 U.S. 697 (1931).

136. *See* Alexander v. United States, 509 U.S. 544, 550 (1993); Mark Deffner, Note, *Unlawful Linking: First Amendment Doctrinal Difficulties in Cyberspace*, 3 MINN. INTELL. PROP. REV. 111, 121 (2002).

137. Hamilton & Sabety, *supra* note 2, at 270 ("By authorizing protection for languages, the [Copyright] Act would be authorizing prior restraint of any expression in that language.").

monopolizing certain speech. The same could hold true for allowing patents on programming tools because granting such patents implies the right to exclude others from writing equivalent programming tools.[138]

Courts have not yet adjudicated whether and to what extent patent law can be the basis for prior restraints. However, prior restraints have been addressed in trade secret cases, in which the alleged infringer threatened unauthorized disclosure of trade secrets and was enjoined from such disclosure.[139] This prohibition of speech can be justified on the basis that trade secret law is content neutral[140] and injunctions based on content-neutral laws are rarely considered prior restraints.[141] The reason why content-neutral injunctions are not usually classified as prior restraints is founded on the rationale that incidental restrictions on speech do not raise the same censorship concerns as restrictions based on laws that are targeted at speech.[142] The same argument would be relevant for speech restrictions as a matter of section 101 allowing patents on programming tools. Because such patents would be based on a content-neutral law, pertinent injunctions would not present prior restraint in violation of the First Amendment.

E. Vagueness and Overbreadth

Due process requires that laws clearly define the activities that are restricted.[143] Two doctrines that can render a speech-restricting law invalid are vagueness and overbreadth. A statute is unconstitutionally vague "if men of common intelligence must necessarily guess at its meaning and will differ as to its application."[144] A statute is unconstitutionally overbroad if it regulates substantially more speech than what the Constitution allows to be regulated. In this regard, section 101 is subject to evaluations of vagueness and overbreadth. However, overbreadth is a "strong medicine," to be employed "sparingly and only

138. *See* 35 U.S.C. § 271(a) (2006).

139. *See generally* Bridge C.A.T. Scan Assocs. v. Technicare Corp., 710 F.2d 940 (2d Cir. 1983); Ford Motor Co. v. Lane, 67 F. Supp. 2d 745, (E.D. Mich. 1999); Religious Tech. Ctr. v. Lerma, 897 F. Supp. 260 (E.D. Va. 1995); DVD Copy Control Ass'n v. Bunner, 31 Cal. 4th 864, 886 (Cal. 2003); Garth v. Staktek Corp., 876 S.W.2d 545 (Tex. App. 1994).

140. *DVD Copy Control Ass'n*, 31 Cal. 4th at 878.

141. *See* Avis Rent A Car Sys., Inc. v. Aguilar, 529 U.S. 1138, 1142 (2000); *DVD Copy Control Ass'n*, 31 Cal. 4th at 886.

142. Thomas v. Chi. Park Dist., 534 U.S. 316, 323 (2002).

143. *See* Grayned v. City of Rockford, 408 U.S. 104, 108 (1972) ("It is a basic principle of due process that an enactment is void for vagueness if its prohibitions are not clearly defined.").

144. Connally v. Gen. Constr. Co., 269 U.S. 385, 391 (1926).

as a last resort."[145] Thus, an overbreadth challenge will not succeed unless the statute "is not readily subject to a narrowing construction."[146] This is true for a finding of vagueness as well.[147] Applying this reasoning to the interpretation of patent-eligible subject matter under section 101, section 101 could be interpreted not to include programming tools. Thus, the statute is not invalid for vagueness or overbreadth.

V. SUMMARY OF RESULTS

Having reached the end of the analysis, what are the results? First, section 101 cannot be interpreted to cover patents on programming languages. Patents on programming languages would effectively be patents on grammars. Those grammars are the actual subject matter to which patents on programming languages refer. They are the elementary building blocks that define the languages. However, grammars do not exhibit sufficient physicality and functionality as required by contemporary patent eligibility tests. Rather they are abstract ideas that are neither new nor useful in the sense of section 101.

However, from a patent law perspective, the patent ineligibility of programming languages is of little practical relevance due to the patent eligibility of language-recognizing programming tools, such as compilers. Compilers are programs for identifying whether source code was written according to the grammars they recognize and converting them into target machine code. Obtaining a patent on a compiler or other recognizer gives the patent holder the same control over the software implementation of a programming language as if its grammars were patent-eligible.

While exclusion of grammars from patent eligibility is of little practical relevance from a patent law perspective, it has substantial advantages from a First Amendment point of view. As source code written in a programming language can be protected speech, holding grammars patent-ineligible prevents patent holders from excluding third parties from an entire language. As it turns out, it is possible to advance free speech under the First Amendment while at the same time promoting the progress of science and useful arts.

145. Broadrick v. Oklahoma, 413 U.S. 601, 613 (1973); Bernstein v. U.S. Dep't of State, 922 F. Supp. 1426, 1438 (N.D. Cal. 1996); Bernstein v. Dep't of State, 945 F. Supp. 1279, 1294-95 (N.D. Cal. 1996).

146. Erznoznik v. City of Jacksonville, 422 U.S. 205, 216 (1975) (citing Dombrowski v. Pfister, 380 U.S. 479, 497 (1965)).

147. *E.g.*, Coates v. City of Cincinnati, 402 U.S. 611, 619-20 (1971).