PATENTING COMPUTER DATA STRUCTURES: THE GHOST, THE MACHINE AND THE FEDERAL CIRCUIT

Andrew Joseph Hollander [1]
Dickstein, Shapiro, Morin & Oshinsky
hollandera@dsmo.com

Courts view "data structures," the mechanism by which computers store data in meaningful relationships, differently than do computer scientists. While computer scientists recognize that data structures have aspects that are both physical (how they are stored in memory) and logical (the relationships among the stored information), the Federal Circuit, in its attempts to set clear standards of the scope of patentability of data structures, has not fully appreciated their dualistic nature. This i-brief explains what data structures are, explores how courts have wrestled with setting a limiting principle to determine their patentability, and discusses the resultant impact on claim drafting.

INTRODUCTION

¶1 You're a contestant on the TV quiz show "Jeopardy!" Host Alex Trebek throws a softball by stating, "This is the continent with the largest area." Confident, you ring in and declare "Asia." "Wrong," Alex replies. Another contestant rings in and answers, "What is Asia?" "Correct" says Alex. In "Jeopardy!", you must respond in the form of a question. If you do not, you lose even if you know the correct response.

¶2 Although the above example trivializes the issue, it suggests how patent law treats data structures. Drawn to a machine or manufacture, data structures have been found to pass muster under 35 U.S.C. § 101. However, when drawn to a method, data structures have been found to be nonstatutory subject matter and unpatentable as an abstract idea. As in Jeopardy!, the data structure in question may be rewarded by the patent laws for its physical manifestation but penalized for its logical conceptualization. For computer scientists, however, the physical and logical aspects of a data structure are two sides of the same coin.

¶3 Obtaining a patent on a data structure depends on which side of the coin the U.S. Patent and Trademark Office ("PTO") sees. The first hurdle that an invention must overcome to be patented is 35 U.S.C. § 101: is the subject matter of the invention of the type recognized by the patent laws? Thomas Jefferson, who authored the Patent Act of 1793, wrote that patentable inventions are those that constitute "any new and useful art, machine, manufacture, or composition of matter, or any new and useful improvement thereof."[2] The current language of 35 U.S.C. § 101 defines statutory subject matter identical to Jefferson's formulation, except that "art" is replaced with "process." Thus, statutory subject matter today comprises a new, useful type of process, machine, manufacture, or composition of matter.

---

[1] M.S. (Computer Science), Fordham University, 2003; J.D., Cornell University, 1991; B.A., Wesleyan University, 1984.

¶4 To lend perspective, these days the PTO rejects very few claims in patent applications for failing to meet § 101. The rejections are usually based on 35 U.S.C. §§ 102 and 103 (does prior art render the claimed invention respectively not "new" or render it obvious) and 35 U.S.C. § 112 (is the claimed invention inadequately disclosed or claimed). The legislative history accompanying the Patent Act of 1952 notes that patentability extends to "anything under the sun that is made by man."[3] In *Diamond v. Diehr*,[4] which affirmed the modern judicial view that computer-related inventions are patentable, the Supreme Court acknowledged as much. In *State Street Bank & Trust Co. v. Signature Fin'l Group, Inc.*, the Court of Appeals for the Federal Circuit further remarked that "the repetitive use of the expansive term 'any' in § 101 shows Congress's intent not to place any restrictions on the subject matter for which a patent may be obtained beyond those specifically recited in § 101."[5] Broad though it might be, § 101 excludes laws of nature, physical phenomena and abstract ideas. "A new mineral discovered in the earth or a new plant found in the wild is not patentable subject matter. Likewise, Einstein could not patent his celebrated law that $E=mc^2$; nor could Newton have patented the law of gravity. Such discoveries are 'manifestations of . . . nature, free to all men and reserved exclusively to none.'"[6] Nor can one patent mere abstract ideas, a critical admonition that courts have made in the context of computer-related inventions.

¶5 Thus, for a patent claim related to a data structure, if drawn to a machine or manufacture—the physical aspect of the data structure—precedent permits the claim to survive § 101. But, if drawn to a process—the logical aspect of the data structure—the data structure may be rejected as an abstract idea. When it comes to data structures, then, the Federal Circuit views the logical "ghost" as essentially unpatentable subject matter, but the "machine" that the ghost animates as patentable subject matter.

¶6 While they may not be entirely familiar with the judicial terrain of data structures, patent attorneys drafting patent claims directed to data structures know that they are on firmer ground when emphasizing the "physical" nature of the data structure and deemphasizing the "logical" nature of data structures. This i-brief explains what data structures are, explores how courts have wrestled with setting a limiting principle to determine their patentability, and discusses the resultant impact on claim drafting.

ROLE OF DATA STRUCTURES

¶7 A key difference between a shopping list scribbled on a scrap of paper and the same shopping list stored in a computer is what makes computers so useful in navigating oceans of information: the computer shopping list is stored by means of a data structure which permits the data to be manipulated.

---

[2] *See* Diamond v. Chakrabarty, 447 U.S. 303, 309 (1980).
[3] S. Rep. No. 82-1979 at 5 (1952); H.R. Rep. No. 82-1923 at 6 (1952).
[4] 450 U.S. 175, 193 (1981).
[5] 149 F.3d 1368, 1373 (Fed. Cir. 1998).
[6] *Chakrabarty*, 447 U.S. at 309 (1980).

¶8 Typical data structures include the array, the stack, the linked list, the tree, and also "classes."[7] Computer scientists have devoted substantial efforts to analyzing these and other data structures and finding successful ways to use them efficiently in practice. Despite these efforts, there is no canonical definition of "data structure." Nevertheless, the plasticity of its definition is rather secondary for computer science because the precise definition of "data structure" is simply not critical to the development and use of data structures. Computer scientists know a data structure when they see one. Unfortunately, in assessing whether the data structures before them are patentable, courts have failed to completely appreciate the nature of data structures in an effort to determine their patentability and, consequently, have become mired in difficult issues of patent law.

PLASTICITY OF THE MEANING OF DATA STRUCTURE

¶9 The problem lies in the ontology of data structures—what "is" a data structure? Is it a relationship among data, or the representation of that data as stored in memory, or both?

¶10 Computer science understands that this question dramatizes the distinction between logical and physical representations of data. As described below, logical representations emphasize data relationships and how the data are related from the view of the computer user. Physical representations emphasize issues of effectively and efficiently storing data in memory. These concepts apply to systems as well. For example, in relational database design one performs "logical database design" before "physical database design."[8] Logical database design refers to specifying user-defined data relationships in "schemas" by mapping those relationships with each other.[9] Physical database design "is the process of choosing specific storage structures and access paths . . . to achieve good performance for the various database applications."[10]

¶11 Logical and physical data representations are inextricably bound. They cannot exist independently. In a world where computers had no physical storage structures, there could exist no logical relationships among data in the first place. And, in a world where computers had only physical storage, but no logical relationships among data, computer use would be an onerous task of bit-by-bit manipulation comparable to

---

[7] Classes, which are at the heart of object-oriented languages, like Java, are recent options that programmers can use to carry out the functions of traditional data structures. Object-oriented programming languages are founded on class inheritance properties. For example, a "dog" class is a blueprint from which a "beagle" object can be made ("instantiated"). The "beagle" object inherits properties common to the dog class, e.g., four legs, likes to chew bones, etc. (One could also make "poodle" or "St. Bernard" objects, each of which would share with the beagle object four legs and a fondness for bone-chewing.) The beagle object can be given beagle-specific data, e.g., "color" = white with black and brown spots, and behaviors, e.g., "play" = follows a trail. All this is possible by means of the class data structure. In the real world, class-based modularization yields speedier software development and facilitates software component reuse, practices promoted by the "class" data structure itself.

[8] RAMEZ ELMASRI, ET AL., FUNDAMENTALS OF DATABASE SYSTEMS 549 (3d ed. 2000).

[9] *Id.*

[10] *Id.*

using an extremely complex electronic abacus. Neither of these counterfactual worlds exists, however, because the logical and the physical share a mutual dependency.

¶12 Nevertheless, many notions of "data structures" ignore the physical representation in favor of the logical. This is understandable. Most computer professionals do not need to bother with specifying the physical storage of data in computer memory. Rather, physical storage considerations, important as they may be, are tasks assigned to high-level programming languages, applications programs and operating systems. In fact, best practices often involve "hiding" physical storage problems from the computer user.

¶13 Consequently, many definitions of "data structure" emphasize logical, not physical, relationships. As but one example, a current computer science text defines "data structure" as "a construct within a programming language that stores a collection of data."[11] A data structure is here seen as a construct *in a programming language*, not its manifestation in memory. Moreover, the notion of a "construct" implies something that is abstract. Many computer science texts, monographs and dictionaries are consistent with this definition and stress the logicality of data structures while deemphasizing or ignoring altogether their physicality.[12]

¶14 Other definitions of "data structure", however, acknowledge the term's duality. They recognize that data structures are data stored in a certain logical way—but in physical memory. Examples of this broader approach are found in the *IBM Dictionary of Computing*.[13] One entry defines a "logical data structure" and a separate entry a "physical data structure." A "logical data structure" is "the relationship among data elements from the point of view of an end user."[14] A "physical data structure" is "the form in which data is stored on storage media."[15] Other authorities recognize that the logicality of a data structure is paired with a physical component.[16]

---

[11] F.M. CARRANO AND J.J. PRICHARD, DATA ABSTRACTION AND PROBLEM SOLVING WITH C++—WALLS AND MIRRORS, p. 115 (3d ed. 2002).

[12] These include the following: JEAN-PAUL TREMBLAY, AN INTRODUCTION TO DATA STRUCTURES WITH APPLICATIONS, 4-5 (1976) ("data structures" are distinct from "storage structures" which are "[t]he representation of a particular data structure in the memory of a computer"); ALFRED V. AHO, JOHN E. HOPCROFT AND JEFFREY D. ULLMAN, DATA STRUCTURES AND ALGORITHMS, 13 (1983); ROBERT SEDGEWICK, ALGORITHMS 30 (1983); ELLIS HOROWITZ AND SARTRAJ SAHNI, FUNDAMENTALS OF DATA STRUCTURES 7 (2d ed. 1982); DANIEL F. STUBBS AND NEIL W. WEBRE, DATA STRUCTURES WITH ABSTRACT DATA TYPES AND PASCAL, 6 (1985); PHILIP E. MARGOLIS, RANDOM HOUSE COMPUTER & INTERNET DICTIONARY 128 (3d. ed. 1999); GREGORY L. HEILEMAN, DATA STRUCTURES, ALGORITHMS, AND OBJECT-ORIENTED PROGRAMMING (1996); THE BRITISH COMPUTER SOCIETY GLOSSARY OF COMPUTING TERMS 254 (9th ed. 1998); DICTIONARY OF COMPUTING 119 (3d ed. 1990) (distinguishing data structure from "storage structure"); MICROSOFT COMPUTER DICTIONARY 126 (4th ed. 1999).

[13] IBM DICTIONARY OF COMPUTING (10th ed. 1994).

[14] *Id*. at 397.

[15] *Id*. at 509.

[16] These include the following: ROBERT J. BARON AND LINDA G. SHAPIRO, DATA STRUCTURES AND THEIR IMPLEMENTATION 1 (1980); C.C. GOTLIEB AND LEO R. GOTLIEB, DATA TYPES AND STRUCTURES 29 (1978); DICTIONARY OF COMPUTERS, DATA PROCESSING AND TELECOMMUNICATIONS 130 (1984).

¶15 As one author of a computer science monograph observed, "[a]ny discussion of data structures is fraught with the potential for confusion, because the term 'data structure' has at least four distinct, but related, meanings."[17] None of the meanings given by this author, however, explicitly builds in the requirement of physical memory for the meaning of data structure. On top of all this, one computing dictionary practically ignores the logical aspect, defining data structure as "[t]he physical layout of data."[18]

¶16 This brief survey of the meanings given "data structure" is not meant to imply that any of the definitions are clearly "right" and others clearly "wrong." Each is a defensible definition depending on context, and there is considerable overlap, although many computer scientists would agree that a full understanding of the term should take into account both its logical and physical aspects. Rather, the survey is meant to reveal that, in the final analysis, the meaning of "data structure" is a plastic, context-dependent notion.

JUDICIAL INTERPRETATION OF DATA STRUCTURES

¶17 This plasticity in the definition of data structures, however, has been the source of judicial difficulties. Data structures, like other aspects of computer-related inventions, have met with controversy when it comes to their patentability. A good introduction to the tricky issues associated with patenting data structures is a tandem of Federal Circuit cases issued within two weeks of each other in 1994: *In re Warmerdam*,[19] where five method claims to data structures were deemed unpatentable though one system claim was found patentable, and *In re Lowry*,[20] where system claims to data structures were found patentable.

¶18 In *In re Warmerdam*, the subject invention was designed to help a robot avoid collision with fixed or moving objects. Each of these objects was deemed to be enclosed by a bubble.[21] If the robot came into contact with the bubble's edge, it represented a collision.[22] To gain greater accuracy, however, each object contained a set of smaller bubbles to enable more granular collision-avoidance calculations.[23] This made for a "bubble hierarchy."[24] Warmerdam's claimed innovation was in locating the bubble hierarchy "along the medial axis of the object."[25]

¶19 The Board of Patent Appeals affirmed the Examiner's rejection of six claims. Claims 1-4 were directed to a "*method* for generating a data structure" (emphasis added).[26] Claim 5 was directed to a "*machine* having a memory which contains data representing a bubble hierarchy *generated by the method* of any of

---

[17] CHRIS OKASAKI, PURELY FUNCTIONAL DATA STRUCTURES 3 (1998).
[18] ALAN FREEDMAN, THE COMPUTING GLOSSARY 96 (8th ed. 1998).
[19] 33 F.3d 1354 (Fed. Cir. 1994).
[20] 32 F.3d 1579 (Fed. Cir. 1994).
[21] *Warmerdam*, 33 F.3d at 1356.
[22] *Id.*
[23] *Id.*
[24] *Id.*
[25] *Id.* at 1356

Claims 1 through 4" (emphasis added).[27] Claim 6 was directed to a "data structure generated by the method of any of Claims 1 through 4" (emphasis added).[28]

¶20 On appeal, the Federal Circuit sustained the rejection of claims 1-4 and 6 for failing § 101 and found claim 5—the sole machine claim—the only patentable claim. What, asked the Federal Circuit, is a "data structure"? As the term was not defined in the specification, the court looked to the *IEEE Standard Computer Dictionary*, which defined it as a "*physical or logical* relationship among data elements, designed to support specific data manipulation functions.'"[29] It is unclear why the Federal Circuit emphasized "physical or logical" in the text. Indeed, that the data structure appears to have a physical component, as required by the IEEE definition, undermines the court's conclusion that the data structure represented "abstract ideas." The physicality of Claim 5 implies a machine-like (or manufacture-like) compliance with § 101.

¶21 Compounding this difficulty, the Federal Circuit attempted to distinguish the bubble hierarchy method claims found unpatentable in *Warmerdam* with certain data structures found patentable in *In re Bradley*.[30] The *Bradley* data structure employed firmware programmed with microcode[31] which the court described as "a physical interconnected arrangement of hardware and thus embraced by the term 'machine.'"[32] Once again, the IEEE definition of data structure as "logical or physical" does not support the Federal Circuit's more limited interpretation of the data structure as physical. Nevertheless, the court did not find the precise definition of "data structure" to be an independent ground for denying patentability.

¶22 Claim 5, the only patentable claim according to the *Warmerdam* court began: "A machine having a memory which contains data representing a bubble hierarchy . . . ."[33] In reversing the Board of Patent Appeals, the court acknowledged that the "machine" of Claim 5 was "presumably a general purpose computer," and held that "Claim 5 is for a machine, and is clearly patentable subject matter" under § 101.[34] Claims 1-4, on the other hand, claimed "[a] method for generating a data structure . . . ."[35] Siding with the appellant's argument, the Federal Circuit noted that claim 1, from which the other three method claims depended, was broad enough to cover methods involving physically measuring an object with a ruler or even "eyeballing" it.[36] Thus, it was not specifically drawn to a mathematical algorithm. And though the court

---

[26] *Id*. at 1357.
[27] *Id*.
[28] *Id*.
[29] *Id*. at 1362.
[30] 600 F.2d 807 (C.C.P.A. 1979).
[31] Firmware is essentially computer code hardwired into a Read Only Memory (ROM) circuit. A typical example is a computer's Basic Input/Output System (BIOS), or that part of the computer that launches the operating system, establishes contact with peripheral components, etc.
[32] *Warmerdam* 33 F.3d at 1362.
[33] *Id*. at 1358.
[34] *Id*. at 1355.
[35] *Id*. at 1357.
[36] *Id*. at 1359.

noted that "the preferred, and it appears the only practical, embodiment of the method involves steps which are essentially mathematical in nature, i.e., utilization of the Hilditch Skeletonization method to locate the medial axis,"[37] it concluded, nonetheless, that "the claim involves no more than the manipulation of abstract ideas", and was thus unpatentable under § 101.[38] Nor did the method claims produce physical activity that precluded a § 101 rejection.[39] As a result, the Federal Circuit affirmed the PTO rejection of these four method claims which did no more than manipulate "abstract ideas" as well as claim 6 because it was simply "another way of describing the manipulation of ideas contained in claims 1-4."[40] The practical lesson of *Warmerdam* is that data structures are patentable under § 101 when directed to a machine, but may not be patentable when drawn to a method of data manipulation.

¶23 Two weeks after *Warmerdam*, the Federal Circuit issued another decision refining the scope of patentability of data structures. In *In re Lowry*,[41] the court had before it an appeal from the Board of Patent Appeals on an application entitled "Data Processing System Having a Data Structure With a Single, Simple Primitive."[42] The claimed invention was directed to "the storage, use and management of information residing in a memory."[43]

¶24 The *Lowry* court understood the meaning of "data structure" as "the *physical implementation* of a data model's organization of the data", where a "data model" was a "framework for organizing and representing information used by an application program" (emphasis added).[44] Thus, a data model was a framework for organizing data, and data structures the means to physically implement the organization. It is interesting that "data structure" was here defined as a physical creature, where in *Warmerdam* it was both physical and logical (and the physical was there deemphasized).

¶25 Lowry's invention, the court believed, blended the best features of "functionally expressive" data models and "structurally expressive" ones.[45] Functionally expressive data models, declared the court, enabled complex nesting operations[46] using large blocks of data, but these models have relatively fewer applications and more complex interfaces. "Structurally expressive data models," on the other hand, contained "more varied data structures capable of representing accurately complex information," but made complex nesting

---

[37] *Id*. at 1360.
[38] *Id*.
[39] *Id*.
[40] *Id*. at 1361.
[41] 32 F.2d 1579 (Fed. Cir. 1994)
[42] *Id*. at 1580.
[43] *Id*.
[44] *Id*.
[45] *Id*.
[46] Like a Russian doll that contains a series of smaller dolls within it, a "nested" operation is one wholly contained by another. A simple example is a "nested if", which nests two or more conditions. The pseudocode for the nested conditions found in a husband's driving decision might be: "if (streetlight is yellow) {if (wife is in labor) speed up; else slow down;}"

operations "quite difficult." Lowry described "attribute data objects" ("ADOs") composed of "sequences of bits which are stored in the memory as electrical (or magnetic) signals that represent information."[47] Claim 1 began: "1. A *memory* for storing data for access by an application program being executed on a data processing system, comprising: a *data structure* stored in said memory . . . ."[48] (emphasis added). The Examiner rejected claim 1 and the four claims dependent on it under § 101 as nonstatutory subject matter.

¶26 The PTO Board of Appeals reversed the Examiner, finding that the claim "recited an article of manufacture" for being "directed to a memory containing stored information"[49] and thus satisfied 101.[50] Only the Examiner's rejections under § 102 and § 103, undisturbed by the Board, were on appeal, yielding some regrettable reasoning by the Federal Circuit.

¶27 The § 102 and § 103 rejections flesh out the Federal Circuit's understanding of data structures, with provocative implications for interpreting data structures under § 101. The § 102 and § 103 rejections were founded on the "printed matter" doctrine, which permits patenting only if there is a "'new and unobvious functional relationship between the printed matter and the substrate.'"[51] The Board had "framed the question as whether a new, nonobvious functional relationship exists between the printed matter (data structure with ADOs) and the substrate (memory)."[52] Finding no functional relationship, the Board determined that Lowry could not distinguish his claimed invention on § 102 or § 103 grounds over the prior art, a U.S. patent that disclosed "a CPU using a memory and containing stored data in a data structure . . . ."[53] In doing so, said the Federal Circuit, the Board "erroneously extended a printed matter rejection under § 102 and § 103 to a new field in this case, which involves information stored in memory."[54]

¶28 As an initial matter, said the Federal Circuit , the "printed matter" rejection "'stands on questionable legal and logical footing'" where, as here, the invention is processed not by the mind but by a computer.[55] In any event, said the Federal Circuit, the Board misunderstood the true nature of Lowry's innovative data structures. But in distinguishing Lowry's data structures from printed matter, the Federal Circuit's approach to data structures was inconsistent with its *Warmerdam* opinion issued just two weeks earlier.[56] The Federal Circuit began: "In Lowry's invention, the stored data adopt *no physical 'structure'* per se. Rather the stored

---

[47] *Id*. at 1580-81.
[48] *Id*. at 1581.
[49] *Id*.
[50] *Id*.
[51] *Id*. at 1582 (quoting *In re Gulack*, 703 F.2d 1381, 1386 (Fed. Cir. 1983)).
[52] *Id*.
[53] *Id*.
[54] *Id*.
[55] *Id*. at 1583 (quoting *Gulack*, 703 F.2d at 1385 n.8).
[56] The two cases were heard by different judges. *Warmerdam* was heard before Judges Plager, Lourie and Clevenger. *Lowry* was heard before Judges Rich, Skelton and Rader.

data exists as a collection of bits having information about relationships between the ADOs. Yet this is the essence of electronic structure."[57]

¶29 Thus, the "essence of electronic structure" is to store bits containing information about "relationships," but the stored data do not have a "physical structure." However, the Federal Circuit then went on to say, "[i]n short, Lowry's data structures are *physical entities* that provide increased efficiency in computer operation" (emphasis added).[58] Because Lowry's innovative data structures were physical and promote computer efficiency, they were "not analogous to printed matter," thus the Board's § 102 and § 103 rejections must be reversed.

¶30 Which is it? Do Lowry's data structures have "no physical 'structure' per se" or are they "physical entities"? Within one page, the Federal Circuit gives seemingly contradictory explanations. This difficulty mirrors that in *Warmerdam* where the Federal Circuit defined a data structure as a "physical or logical relationship among data elements."[59] As a matter of computer science, the Federal Circuit was right on both counts: data structures have logical and physical aspects. Unfortunately, this complicates things as a matter of patent law, as the following shows.

¶31 A "data structure" is context-dependent. For example, consider a simple array data structure containing the numbers 1, 2 and 3 called firstThree[]. The first position ("element") in the data structure firstThree[] contains the "1," second position the "2," and third position the "3." Assume that the memory reserved for firstThree[] contains ten free positions with which to store the three-position list. Thus, the list can occupy any of *physical* memory locations 0-2, 1-3, 2-4, 3-5, 4-6, 5-7, 6-8, or 7-9.[60] It can start at any of *physical* locations 0-7. However, the *logical* relationships of the data are fixed no matter where the data is stored physically: "1" always immediately precedes "2" by one position, and "2" always immediately precedes "3" by one position – no matter whether "1" occupies position 0, 1, 2, 3, 4, 5, 6 or 7.

¶32 It can get even trickier in trying to understand the precise relationship of the logical to the physical aspect of data structures. The computer may not have stored the logically related items adjacent to each other in physical memory. As mentioned above, memory tasks are typically hidden from the user. In other words, for firstThree[], the "1" and "2" may be stored adjacent to each other, but "3" may be stored non-adjacently and many bytes away in memory. Still, the programmer sees the relationship logically, no matter where the "3" is stored physically: it remains 1, 2, 3. Thus, the firstThree[] array, on its face sequential as a logical matter, may in physical memory be scattered.

¶33 Trickier still is the common use of "pointers" in a language like C++, and their use in the data structures represented by linked lists. A linked list is a sequential collection of structures, connected or

---

[57] *Warmerdam* 33 F.3d. at 1583.
[58] *Id*.
[59] *Id*. at 1362.
[60] By convention, an array of n elements begins at element 0 and ends at element n-1.

"linked" by pointers. Linked lists are more flexible than arrays for holding lists of items. A linked list can grow as necessary while an array is limited to the fixed number of elements initially declared for it. A pointer contains a memory address of a variable; the variable contains a specific value. For example, assume the value of a variable named "temperature" is "98.6." A pointer to "temperature" contains not 98.6 but a memory address, e.g., 0x0066FDEC.

¶34 To illustrate, assume that the data items in firstThree[] are stored in a linked list data structure. "1," "2" and "3" can be stored in three separate physical memory locations, linked by a pointer that "remembers" where to access them. The linked list thus appears as a logical sequence.

¶35 Because pointer-based manipulation is essentially use of the physical aspects of data structures (memory) to organize the logical aspects of data structures (the relationships), it further exposes the duality of the meaning of "data structure." In other words, the array data structure is different from a linked list data structure because the programmer deals only with the array positions and the computer hides all memory aspects. With linked lists, on the other hand, the programmer wants to deal with some memory aspects—pointers. Thus, the linked list data structure, by its very nature, has a physical aspect beyond the logical: memory address manipulation, by pointers that point to memory addresses. Even so, the programmer does not directly manipulate the specific memory address in the hardware (e.g., 0x0066FDEC). Instead, the programmer declares a pointer variable that points to a stored address.

¶36 In light of the discussion above, it is understandable why the Federal Circuit's views of "data structure" in *Warmerdam* and *Lowry* cannot easily be harmonized. A data structure can have logical properties, or as *Lowry* put it, be "a collection of bits having information about relationships."[61] This is analogous to the *logical* relationship of "1" to "2" – always preceding it by one position, no matter where stored. And, a data structure can have physical properties, or as *Lowry* put it represent "physical entities" with "specific electrical or magnetic structural elements in a memory."[62] This is analogous to the *physical* storage of "1" at either position 0, 1, 2, 3, 4, 5, 6 or 7 in an array.

IMPACT ON CLAIM DRAFTING

¶37 While *Warmerdam* and *Lowry* reveal that a physically understood data structure can satisfy § 101 as drawn to a machine or manufacture, *Warmerdam* reveals that a logically understood data structure expressed in method claims can be rejected under § 101 as an "abstract idea." Yet, in the computer program employing it, the data structure may have a single resulting functionality. The *raison d'être* of computers, after all, is their functionality, i.e., what a computer provides the end-user. So whether data structures are understood in

[61] *Warmerdam*, 33 F.3d at 1583.
[62] *Id*. at 1583-4.

logical terms, or physical terms, or both, is an important, but secondary, question. First and foremost is functionality: what data structures can do.

¶38 Inasmuch as *Warmerdam* found essentially the same data structure unpatentable when drawn to a method claim, but patentable when drawn to a machine claim, the physical/logical duality of a data structure can result in strategic claim drafting. Emphasizing the physical nature of a data structure, one might draft a claim directed to an article of manufacture or machine because if characterized as a method, the claim may not survive § 101.

¶39 Indeed, PTO procedures encourage as much. Effective March 29, 1996, the PTO amended the detailed Manual of Patent Examining Procedure ("MPEP"), a *vade mecum* for Examiners directed to all areas of the technical arts, to include section 2106, commonly known as the Examination Guidelines for Computer-Related Inventions. In particular, the PTO's MPEP rewards claiming software as an article of manufacture. Specifically, "a claimed computer-readable medium encoded with a data structure defines structural and functional interrelationships between the data structure and the computer software and hardware components which permit the data structure's functionality to be realized, and is thus statutory."[63] In contrast, "[d]ata structures not claimed as embodied in computer-readable media are descriptive material per se and are not statutory because they are not capable of causing functional change in the computer."[64] This has become known as a so-called "Beauregard claim," after PTO proceedings referenced in *In re Beauregard*.[65] Thus, software claimed as an article of manufacture—that is, software embodied on a computer-readable medium—is patentable.

¶40 Consequently, it is hardly surprising that a November 2003 search of the PTO database reveals that the phrase "computer-readable medium" appears in the claims of over 5,500 issued U.S. patents. Many if not most presumably cover software expressed as an article of manufacture or a machine, two of the four statutory categories satisfying 35 U.S.C. § 101 of the patent laws.[66]

¶41 Claims define the legal scope of the invention. Indeed, "[t]he name of the game is the claim."[67] Some believe that strategic claim drafting exalts form over substance, where claim language can confer patentability on data structures as understood physically, but not logically. To the extent that the courts or

---

[63] MPEP § 2106(IV)(B)(a).
[64] *Id.*
[65] 53 F.3d 1583 (Fed. Cir. 1995).
[66] For example, U.S. Patent No. 6,651,243, issued Nov. 18, 2003, provides, as claim 31:
"A computer program product in a *computer-readable medium* for use in a data processing system for profiling an executing program, the computer program product comprising:
first instructions for performing sample-based profiling of the executing program;
second instructions for generating, for each sample period, a *tree data structure* in which nodes of the tree data structure represent routines of the program that execute during the sample period; and
third instructions for merging, in response to a determination of completion of the execution of the program, tree data structures from each sample period into a resulting tree data structure." (emphasis added).
[67] In re Hiniker Co., 150 F.3d 1362, 1369 (Fed. Cir. 1998).

Congress wish to refine standards for software patentability to minimize strategic claim-drafting, they will need to appreciate properly the plasticity of the meaning of “data structures.”