Should Program Algorithms be Patented?

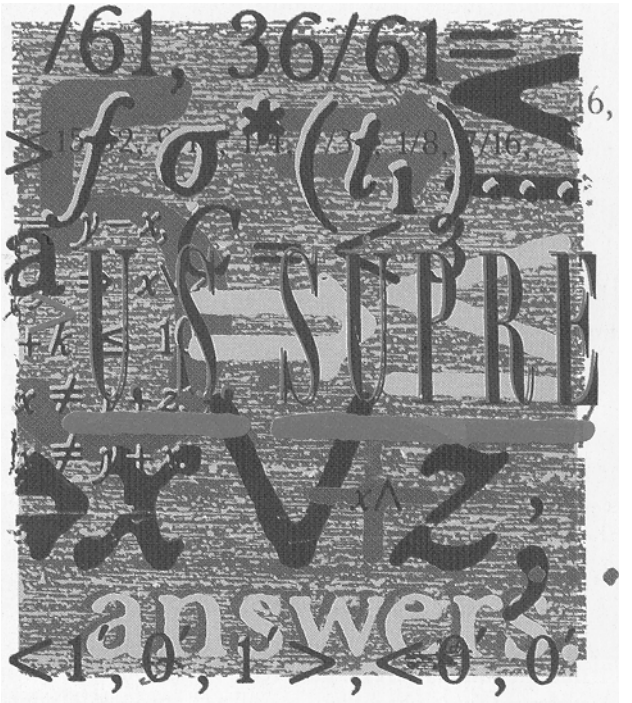

ILLUSTRATION - DENNIS AHLGRIM

In the *Legally Speaking* column last May [6], we reported on a survey conducted at last year's ACM-sponsored Conference on Computer-Human Interaction in Austin, Tex. Among the issues about which the survey inquired was whether the respondents thought patent protection should be available for various aspects of computer programs. The 667 respondents overwhelmingly supported copyright protection for source and object code although they strongly opposed copyright or patent protection for "look and feel" and most other aspects of programs. Algorithms were the only aspect of programs for which there was more than a small minority of support for patent protection. Nevertheless, more than half of the respondents opposed either copyright or patent protection for algorithms. However, nearly 40 percent of the respondents regarded algorithms as appropriately protected by patents. (Another eight percent would have copyright law protect them.)

We should not be surprised that these survey findings reflect division within the technical community about patents as a form of protection for this important kind of computer program innovation. A number of prominent computer professionals who have written or spoken about patent protection for algorithms or other innovative aspects of programs have either opposed or expressed reservations about this form of protection for software [2, 4, 5].

This division of opinion, of course, has not stopped many firms and some individuals from seeking patent protection for algorithms or other software innovations [8]. Although the Refac Technology patent infringement lawsuit against Lotus and other spreadsheet producers may be in some jeopardy, it and other software patent lawsuits have increased awareness of the new availability of software patents. This situation, in turn, has generated some heated discussion over whether this form of legal protection will be in the industry's (and society's) long-term best interests.

The aim of this column is to acquaint readers with the legal debate on patent protection for algorithms and other computer program innovations, an issue which seems to be as divisive among lawyers as those in the computer field. [3, 9].

The Legal Debate

There are three U.S. Supreme Court decisions that seem to state quite plainly that computer program algorithms are not the sort of innovation that can be patented. (More precisely, it has been said that program algorithms are not among the kinds of "processes" Congress intended to be eligible for patent protection when it passed the patent statute.) But as Ecclesiastes once said, "God made mankind straight, but men have recourse to many subtleties." Patent lawyers have found ways of interpreting these three decisions more narrowly than their plain meaning suggested was appropriate; through clever drafting of patent applications they have persuaded the patent office that these decisions do not bar patents for their clients' software innovations.

Some patent lawyers, for example, have interpreted the third of these three Supreme Court decisions as meaning that algorithms are now unpatentable only if no practical

BY
Pamela Samuelson

application is claimed for them [9]. This reading of the judicial opinion ignores too much of the rest of what the Court said in that case to be a convincing interpretation. It also runs counter to a set of guidelines that the Patent & Trademark Office (PTO) issued within the past year concerning the standards by which it would judge patent claims for algorithms [10]. Nevertheless, some recently issued patents suggest that at least some patent examiners are operating on this basis. A lawyer who takes an aggressive stand on the patentability of software innovations is certainly more likely to generate more business for him- or herself than one who has a more cautious interpretation of the patentability standard.

While lawyers have been arguing for many years about the patenting of software innovations, the legal debate over the patenting of algorithms intensified when in 1986 Donald Chisum, an articulate and well-respected patent scholar, wrote an article arguing that the Supreme Court rulings against patent protection for computer program algorithms be overruled. He asserted that the rulings were an incorrect interpretation of patent law as well as being bad intellectual property policy [1]. Since then, the PTO has issued some well-publicized patents for computer program algorithms, including one for industrial applications of Narendra Karmarkar's linear programming algorithm, assigned to AT&T. And this past November, an appellate court overturned a decision by the PTO which would have denied a patent to a voice recognition algorithm. The implications of this case, however, are far from clear, for the decision said it was following the earlier Supreme Court rulings, and upheld the algorithm patent claim on grounds that are far from convincing and seem at odds with at least one of the previous Supreme Court decisions (see following discussion). And so the legal debate continues.

While this column can give only a brief glimpse of the history of the legal debate on this topic [7], it is well to begin with the story of the first computer program algorithm case to be decided by the Supreme Court. This case is typical of the problems that computer program innovations present for the patent system, and it is the case which Professor Chisum has argued should be overruled.

The Benson Case

Gottschalk v. Benson is the 1972 Supreme Court decision that ruled a computer program algorithm is by its nature unpatentable. Gottschalk was the Commissioner of Patents who sought Supreme Court review of an appellate court ruling which had overturned the patent office's decision to deny Benson (an employee of Bell Laboratories) a patent on his two claims for a new algorithm for converting binary-coded decimals to pure binary form.

The appellate court regarded the first of Benson's two claims as easily meeting the standards for a patentable process because the claim made reference to hardware elements, such as "signals" and "reentrant shift registers." This meant, said the judges, that it was only a claim for the machine implementation of this process. Under standards this court had announced in previous cases, such hardware references made the claim a patentable one. The judges pointed out that cash registers, like Benson's method, worked with numbers, but that did not make such registers unpatentable. (This analogy, however, misses the deeper question of whether addition itself would be patentable as a process merely because it is capable of being carried out on a machine such as a cash register, an issue which will be discussed further here.)

Benson's second claim, however, made no mention of any hardware elements. The appellate court admitted that issuing a patent on this claim would cover the method when performed manually with a pencil and paper (which was why the patent office regarded it as an unpatentable "mental process"). Since the court believed computer implementations would be the only practical utilization of the invention, it decided this claim too was technological enough in character to be a patentable process. Consequently, the appellate court ruled that the patent office had been wrong to reject Benson's patent application.

The Supreme Court reversed this appellate court decision and ruled that the patent office had been right to reject both of Benson's claims. It agreed with the patent office that until that time, only processes that involved the transformation of matter from one physical state to another (such as a chemical process) had been considered patentable. Benson's method did not transform matter.

While the court made clear it was not saying that transformation of matter would always be required to support the patentability of a process, the judges were persuaded by "friend of the court" briefs submitted by such firms as IBM, Burroughs, and Honeywell that the mathematical character of the Benson algorithm excluded it from being the sort of process that was patentable in nature. (The Court also agreed with the patent office that mental processes are not patentable, although this was not one of its main points.) The court compared Benson's algorithm to a law of nature or a scientific principle—discoveries traditionally not considered to be patentable in character. The fact that the only practical utilization of the Benson algorithm was in a computer was interpreted by the Court to mean that a patent on it would, in effect, preempt all uses of that algorithm. This factor helped to influence the Court to deny its patentability.

Post-Benson Patentability Standards

In the years that followed the Supreme Court's 1972 Benson decision, the appellate court reviewed a number of other patent office decisions involving computer program innovations. In these cases, the court experimented with various inter-

pretations of the Benson decision (which the appellate court was bound to follow, even if the judges did not agree with the Supreme Court's ruling).

For a while, the appellate court interpreted Benson as applying only to claims drafted in *process* (or method) form, and not to claims drafted in *machine* (or apparatus) form, although a patent lawyer could, through minor wording changes, easily draft the claims in either form. At some point, however, the appellate court decided to abandon this distinction. (But see the following discussion of the Iwahashi case.)

Then the appellate court began to distinguish between "mathematical algorithms" (by which the appellate court generally meant mathematical formulae), which it said were unpatentable under the Benson ruling, and nonmathematical algorithms (such as an algorithm for converting written texts from one natural language to another, which the appellate court regarded as nonmathematical in character), which could be patented.

For a time, the appellate court decided that even claims for "mathematical algorithms" might be patentable as long as the claims did not cover all uses of the algorithm: limiting the claim to some technological environment or field of application was regarded by the appellate court as saving the claim from Benson's proscription against a patent on an algorithm.

However, in 1978 (and again in 1981), the Supreme Court said that a claim limitation of this sort was not consistent with its ruling in Benson. Nor was it consistent with Benson to merely tack on to the claims some minor post-solution activity. In its 1981 decision—Diamond v. Diehr—the Supreme Court ruled that a patent claim for a process should not be rejected merely because it includes a mathematical calculation or a computer program as an element.

The only requirement, the Court said, was that the process being claimed—in Diehr, the process was said to be one for curing rubber, which included as an element some computerized calculations to determine when the curing was done—be of a patentable sort. Since rubber curing is a traditional type of industrial process (i.e., one involving the transformation of matter), the Court found Diehr's process to be patentable in nature. The present PTO guidelines on the patentability of claims involving mathmematical algorithms attempt to implement the Supreme Court's ruling in Diehr, as well as to be consistent with other appellate courts' rulings on computer program-related inventions.

The court compared Benson's algorithm to a law of nature or a scientific principle—discoveries traditionally not considered to be patentable in character.

The Iwahashi Case

Since the Supreme Court's decision in Diamond v. Diehr, there have been relatively few court decisions concerning the patentability of computer program algorithms or other software innovations. In the fall of 1989, however, the appellate court which oversees the patent office's decisions issued two opinions concerning what the PTO found to be unpatentable algorithms. In the Iwahashi case, the appellate court ruled that a patent should have issued; in the Grams case, the appellate court upheld the PTO's rejection of the claims.

Iwahashi's claim was drafted in apparatus (rather than method) form, and was for an auto-correlation unit useful in pattern recognition (particularly voice recognition) to obtain auto-correlation coefficients for stored signal samples. Iwahashi claimed to have invented a simpler way to obtain the desired coefficients. (Rather than utilizing multiplication as the prior art did, which required more complicated circuitry and more calculation time, Iwahashi's unit squared the sum of two factors in accordance with a stated formula.)

Most of the elements in the claim were for obtaining input values, calculating sums in accordance with a formula, and storing the values obtained from the calculations. Several of the claims elements made reference to "read only memory" (e.g., storing a value in read only memory). Despite these references, the PTO regarded the claim as being for the algorithm. The appellate court, however, focused on the fact that the claim was for an apparatus (a "unit"), and made references to read only memory (a hardware component) in ruling that the claim was for a patentable machine.

Given that the Supreme Court, in the course of judging the patentability of Benson's invention, did not distinguish between the claim which referred to "reentrant shift registers" and that which made no reference to hardware elements, the appellate court's ruling in Iwahashi seems inconsistent with Benson.

The Iwahashi opinion does not seem to be in agreement with earlier decisions by a predecessor appellate court, which regarded as immaterial whether a claim reciting a mathematical algorithm was drafted in method or apparatus form. Based upon the reading of this case, one wonders whether all it takes now to render a claim for a computer pro-

gram-related innovation patentable is to draft it in apparatus form and mention a ROM.

Although the decision does not indicate whether the algorithm was intended to be embodied in a program or in a chip, this distinction too would not seem to be meaningful since the Benson algorithm, like all other computer program algorithms, could have been embodied in a chip, rather than a program.

Another computer program algorithm patent which does not appear consistent with the three Supreme Court decisions on this subject, is AT&T's patent on "industrial applications," of the Karmarkar algorithm, in view of the Court's statements that merely limiting the field of application for the algorithm does not make it patentable.

The Grams Case

Grams made a number of claims related to a method of diagnosing abnormal conditions in complex systems. The method consisted of steps such as conducting tests on individual instances of the system, taking values from these tests and comparing them with values associated with normal individuals, and conducting successive tests to determine the cause(s) of the abnormality. The patent application made evident that the method was to be computerized. Relying on the 1982 Meyer decision in which an algorithm for an expert system program for diagnosis of neurological conditions had been held to be unpatentable because it was for a mathematical algorithm, the appellate court in Grams upheld the PTO's rejection of the claims.

One of the surprises about both the Grams and the Meyer decisions was that in each of them the court took a broader view of what the term mathematical algorithm included than it had in some of its earlier decisions. In the 1978 Toma case, for example, the appellate court had rejected the argument that Toma's algorithm for a computerized process of natural language translation was a "mathematical algorithm" for it recited no equation; but then neither did Grams's or Meyer's applications. In the latter two cases, the appellate court also emphasized that the claims were for an unpatentable mental process, even though it was clear that the intended implementation of both was a computer program.

What to do If Patent Law's Distinctions are Untenable

Chisum has argued that the Supreme Court's Benson decision should be overruled. Benson's algorithm was, in his view, a process that was technological enough in character to be patentable. Chisum has blamed the analytic confusion reflected in the judicial case law (such as the distinction between mathematical and nonmathematical algorithms) on the Supreme Court's Benson decision, and predicts that this confusion will be resolved once Benson is overruled. Chisum has also asserted that patent incentives are needed to stimulate investment in research that will lead to important algorithmic innovations and advance the state of the art of computer programming.

One of the surprises about both the Grams and the Meyer decisions was that in each case the court took a broader view of what the term mathematical algorithm included than it had in some of its earlier decisions.

The computer scientist Allen Newell, in responding to Chisum's article on the patentability of algorithms, agreed with Chisum that the distinction between mathematical and nonmathematical algorithms is untenable, as is that between algorithms and mental processes. Newell pointed out that cognitive scientists have been aiming to model the computational processes which occur in the brain by writing programs that stimulate this kind of computation; consequently, there is an equivalence between algorithms and mental processes that makes any distinction between them for patent purposes doomed to failure.

While agreeing with Chisum that the particular confusion that developed in the law in the aftermath of the Benson decision might disappear if Benson were overruled, he questioned Chisum's conclusion that all the analytic confusion in patent law concerning algorithms would be resolved by this act. Newell thought more profound issues were raised by the patenting of program algorithms than Chisum seemed to realize.

Newell suggested that the conceptual models on which the patent system was based might be broken when applied to algorithms and other program innovations, and questioned whether more innovation in program algorithms would result from patenting than has resulted from what has been the norm of nonprotection.

Newell used the example of the commonly used algorithm for addition to illustrate the conceptual problems presented by patents for algorithms. Suppose this example (or some other mathematical procedure of equally widespread application) had just been invented. Chisum's definition of a patentable process would seem to include such an innovation as a patentable one. Yet it is surely the kind of innovation which would ordinarily not be considered patentable. Even the Supreme Court justices who would have upheld a patent on an equation useful in catalytic conversion plants in the 1978 Parker v. Flook case gave

multiplication as an example of an unpatentable process.

The Need for a Standard of Patentability

Before overruling the Benson decision, it is surely wise to think carefully about the consequences of granting patent protection to computer program algorithms, which are mathematical in character. What makes them technological enough to be patentable processes? The fact that they can be carried out on a computer? Some case law suggests this may be enough.

If this is the case, an algorithm for addition would seem to be patentable, as would the program to carry it out. Since a patent could issue on the algorithm itself, and since patent law gives the holder of the patent exclusive rights to use the patented invention, some might even argue that it would infringe upon the patent to write an article about the algorithm, for writing about it would use it, inducing one's readers to use it as well; this might be regarded as contributory infringement. Traditionally it has not been an infringement of a patent to draw a patented machine or to write an article about it, for one could not thereby make or use the machine, whereas with mental processes like addition, one can use the invention by writing about it. Computer program innovations, if and when patented, mark the first time it can infringe a patent to embody the innovation in a copyrighted written text.

Given that all types of information can now be processed by computer, a standard of patentability that rested merely on the ability to computerize it would make all methods of representing, organizing, and manipulating information patentable. (Benson's algorithm, for example, can be characterized as a method of representing data—in that case, numerical information—or converting its representation from one form to another.) In the past, methods of representing, organizing, and manipulating information have been considered unpatentable—not technological in character. It seems a rather broad stretch to make all information processing patentable just because one wants, for example, to give AT&T incentives to invest in research for advances in computing such as the Karmarkar linear-programming algorithm. Yet where does one draw the line?

Chisum punts when it comes to indicating what the bounds of patentability would be if Benson were overruled, looking only to a 1970 case which says all it takes to make a process patentable is that it be "in the technological arts," without defining what that might and might not include. Later cases interpreting that 1970 case seem to say that having the ability to be carried out on a computer is enough to make a process patentable. Transformation of matter as a test of what patentable processes might include and not include may have outlived its usefulness, but at least it was a standard that provided some limiting boundaries for patentability.

Furthermore, considering how the software industry has grown, and the amount of innovation it has exhibited in an environment in which patent protection was perceived to be unavailable, some have questioned whether more than copyright or trade secret protection is really needed to provide incentives for innovations in computer programming.

Some also express concern about the ability of the PTO to make up for 30 years of not keeping track of the state of the art of computer programming, and the adequacy of its classification system, as well as its judgment as to the nonobviousness of some innovations which have been patented. In addition, some worry that the structure of the software industry will be changed by the increasing utilization of patents for software innovations, and entry into the business will be made more difficult. Considering how much innovation in the field has come from small firms, the prospect of higher entry barriers from patents is worth considering carefully.

A New Policy Study Is Underway

At the request of some congressional committees, the Office of Technology Assessment (OTA) of the U.S. Congress has just recently undertaken the study of how the U.S. software industry can most effectively utilize intellectual property protection to maintain its competitive edge in the emerging global marketplace for software. Among the issues OTA will study is what role patents should play in the legal protection of program innovations. OTA will be seeking input from computing professionals, industry, user groups, as well as lawyers, in carrying out this work. OTA may well conclude that although growth of the industry might have been impeded if patents had been available for program innovations in the early stages, such protection is now needed to spur investment in software development and strengthen the position of U.S. firms in the international arena. What do you think? ◘

Pamela Samuelson is a professor of law at the University of Pittsburgh School of Law.

References

1. Chisum, D. The Patentability of Algorithms. *Univ. of Pittsburgh Law Rev. 47*, 959 (1986).
2. Kapor, M. Testimony at Hearings before U.S. House of Representatives, Subcommittee on Courts, Intellectual Property and the Administration of Justice, of the Committee on the Judiciary (March 5, 1990).
3. Kahin, B. The Software Patent Crisis. *Tech. Rev., 93*, 52 (April 1990).
4. Newell, A. The Models Are Broken! The Models Are Broken. *Univ. of Pittsburgh Law Rev. 47*, 1023 (1986).
5. Plauger, P.J. Soup or Art? (Copyright Protection for Software Concepts), *Comput. Lang. 6*, 17 (Sept. 1989)
6. Samuelson, P. and Glushko, R. Survey on the Look and Feel Lawsuits. *Commun. ACM 33*, (May 1990), 483.
7. Samuelson, P. Benson Revisited: Should Patent Protection Be Available for Algorithms and Other Computer Program-Related Inventions? *Emory Law J., 39*, To be published—fall 1990.
8. Soma, J. and Smith, B. Software Trends: Who's Getting How Many of What? 1978 to 1987, *J. of Pat. & Tradem Soc. 71*, 415 (1989).
9. Sumner, J. and S. Lundberg. The Versatility of Software Patent Protection: From Subroutines to Look and Feel. *Comput. Lawyer, 3*, 1 (June 1986).
10. U.S. Patent & Trademark Office. Report on Patentable Subject Matter: Mathematical Algorithms and Computer Programs. *Pat., Cop., & Tradem. J. (BNA), 38*, 563 (1989).