Constructive Mathematics and Computer Science

Henry Cheng, New Mexico State University

This paper gives an informal exposition of the relationship between mathematics and computer science generated by the constructive viewpoint of Errett Bishop. Brouwer's insight on the lack of computational content in classical mathematics is discussed first. Then Bishop's constructivism is presented as a natural completion on the program started by Brouwer to develop a more realistic foundation for mathematics. Finally, the correlation between formal constructive mathematics and computer languages is illustrated and examined.

KEY WORDS AND PHRASES: informal constructive mathematics, formal constructive mathematics, principle of the excluded middle, intuitionism, data types, implication.

CR CATEGORIES: 5.20, 5.21, 5.23

Both mathematics and computer science are undergoing vigorous development. However, if there is any interaction between these two sciences, it is more on an incidental basis than on any systematic one. This is not a healthy situation and I hope to illustrate a genuine alternative to this status quo. I hope to show that if mathematics is approached from the constructive viewpoint of Errett Bishop, then it is recognized that the development of computer science, at least the software, is essentially a mathematical one. On the other hand, a study of computer languages can be used to delineate the scope of constructive mathematics. To explain the constructive viewpoint, it appears best to contrast it with the classical viewpoint (first enunciated by Plato (1)) that currently dominates mathematical thinking.

Consider the set of all integers Z. Now the classical mathematician or the Platonist presupposes that the entire infinite set Z pre-exists in some immutable world of ideas and that our knowledge or ignorance of Z in no way affects its existence. In a sense, the totality of integers transcends our own existence. There is a most beautiful statement of this attitude by the British mathematician G. H. Hardy (2; p. 63): "I believe that mathematical reality lies outside us, that our function is to discover or observe it, and the theorems that we prove, and which we describe grandiloquently as our 'creations', are simply our notes of our observations. This view has been held, in one form or another, by many philosophers of high reputation from Plato onwards, and I shall use the language which is natural to a man who holds it." The language that Hardy refers to is, of course, the language of Aristotelian logic. And Hardy is absolutely correct when he said that Aristotelian logic is natural to the classical viewpoint of mathematics. The clearest example of this claim is the Aristotelian principle of the excluded middle: for any mathematical statement S, either S is true or S is false. Clearly, if all mathematical ideas, along with the integers, pre-exist in some nebulous world, then their truth values must also have been predetermined, one way or the other. The principle is most often used in the form of double negation: for any mathematical statement S, S is true if and only if not(not S) is true. To show that a continuous function $f:[0,1] \to [0,1]$ has a fixed point, it suffices to show that the assumption that such a function f does not have a fixed point leads to a contradiction. There is no need to even determine the fixed point to within 1/2. So, from the classical viewpoint, mathematics, at least in principle, need not have anything to do with computations. Its sole task is to specify the truth values of certain statements in the metaphysical world of ideas. (It goes without saying that this is very detrimental to the computer scientist.)

The Dutch mathematician Brouwer was one of the first to seriously object to the lack of computational meaning in the classical viewpoint, and named the principle of the excluded middle as the main culprit (3). He considered first a finite sequence $X_1,\ldots,X_n$ of integers and observed that the principle of the excluded middle implies that either $\forall k(X_k \le 0)$ or $\exists k(X_k > 0)$. Brouwer did not quarrel with this application of the principle of the excluded middle because it is always possible in a finite number of simple arithmetical operations to determine one of the two above alternatives. However, Brouwer felt the situation is dramatically different when we consider an infinite sequence $\{X_1,X_2,\ldots\}$ of integers, which are generated by some algorithm. Then in general we have <u>no finite</u> computational method to assert either $\forall k(X_k \le 0)$ or $\exists k(X_k > 0)$. A typical example is the sequence $\{X_k\}_{k=1}^{\infty}$ defined by $X_k = 0$ if 2k is the sum of two primes and $X_k = 1$ otherwise. The case $\forall k(X_k \le 0)$ is Goldbach's conjecture; and at present, we have no proof of Goldbach's conjecture nor have we exhibited a counter-example. Hence the principle of the excluded middle, at least when applied to infinite sets, becomes more an affirmation of faith in that world of ideas than a basic principle of numerical computation. Brouwer thought this was such a blot on mathematics that he set about himself the task of removing the Platonic world of ideas as the basis of mathematical thought and called his program intuitionism. To begin with, he simply denied the <u>existence</u> of the infinite, relegating it to a figment of our imagination. (We should mention that this idea had already been proposed by Kronecker, Poincaré and others.) Brouwer declared that one can speak of an infinite sequence of integers only in the sense of a finite process that can be continued indefinitely, capable of surpassing each finite limit. He felt that mathematics should be based on our intuition, since it is only a human endeavor.

Brouwer's critique of classical mathematics was grudgingly accepted by some of his contemporaries. Even the staunchest defender of classical mathematics yielded ground: David Hilbert stipulated that only finitistic methods are admissible in meta-mathematics (4; p. 62). As to mathematics itself, Hilbert intended to extricate it from the yoke of Brouwer's objections by an appeal to the formal consistency of the classical mathematical system. Due to Gödel intervention, we know now that Hilbert did not achieve his objective. Nonetheless, few mathematicians became advocates of intuitionism. The reason is quite simple. Despite his great contribution in pointing out the deficiencies of classical mathematics, Brouwer failed to provide a viable substitute for the classical viewpoint. His introduction of such ideas as free choice sequences make intuitionistic mathematics no less ethereal (nor more computational) than classical mathematics.

It remained for the American mathematician Errett Bishop to complete the grand design of Brouwer, but from a different viewpoint; namely, constructivism. Instead of basing mathematics on what is intuitively evident, as Brouwer did, Bishop states as his basic constructivist goal (5): ". . . that mathematics concerns itself with the precise description of finitely performable abstract operations. It is an empirical fact that all such operations reduce to operations with the integers . . . . Thus by 'constructive' I shall mean a mathematics that describes or <u>predicts</u> the results of certain finitely performable, albeit hypothetical, computations within the set of integers."

As to the nature of the integers, Bishop agrees completely with Brouwer: leave it to our intuition, and do not derive the integers from some formal system.

Hence Bishop accepts the algorithm

$$X_0 := 1$$
$$X_{n+1} := \frac{1}{3}\left(2X_n + \frac{2}{X_n^2}\right)$$

as a finitely performable sequence of operations and the theorem

$$\lim_{n\to\infty} X_n = (2)^{1/3}$$

as a prediction on the sequence $\{X_n\}$. Note that Bishop never assumes that the sequence $\{X_n\}$ is ever given in its entirety, nor does he ever need this assumption.

From his constructive viewpoint, Errett Bishop has successfully recast in his book, Foundations of Constructive Analysis, a large portion of modern abstract analysis, including the central theory of measure. (This was an accomplishment that David Hilbert did not think was possible.) Bishop does not negate all the achievements of classical mathematics. What he did in his book was to extract and whenever necessary to implant computational meaning in the theorems of classical mathematics. For instance, constructive measure theory is based on a lemma that is classically a generalization of Cantor's diagonization process (6; p. 160). This lemma was later extended to give an elegant measure theory using the ideas of Daniell (7). Moreover, constructive measure theory seems sufficient as a vehicle to study probability (8) and ergodic (9) theories. More examples can be cited; but the important thing is that, in one stroke, Bishop has dispelled the myth that any mathematical system without the services of the principle of the excluded middle will be meager in results and unnaturally cumbersome.

Clearly, the success of Bishop's program will have profound implications on computer science, because what are computers except fast, reliable and versatile machines dedicated to finitely performable operations on the set of integers.

In fact, constructive mathematics appears to be branching into two roles. On the one hand, constructive mathematics will continue to develop on an informal basis. The main concern will be "the communication of algorithms [and predictions about algorithms], with enough precision to be intelligible to the mathematical community" (10). On the other hand, the movement to formalize mathematics is just beginning. Again Errett Bishop has provided the initial impetus. "Formal constructive mathematics is concerned with the communication of algorithms [and predictions about algorithms] with enough precision to be intelligible to machines (10). Of course the machines we have in mind are the computers.

Let us now consider the kind of mathematics that machines or digital computers (I will use these two words interchangeably) are capable of doing or understanding. I should say at the outset that it appears every aspect of our intuition about the integers has a counterpart in the hardware capabilities of the machine. The most basic intuition of counting

$$1, 1+1, \ldots, k, k+1, \ldots$$

has the counterpart of the bit adder. Of course we assume that the bit adder has the property that if an integer k+1 is too large for one register, the overflow will simply be taken up in the next register. Someone may object that although we comprehend the integer

$$X = \left[\left[[10!]^{10!}\right]^{10!}\right]^{10!}$$

a machine certainly cannot store it as a binary number in its memory, because this integer is larger than the total number of molecules in our observable universe. But this is an unfair comparison. To require that this number be stored in its binary representation is equivalent to requiring that we, as humans, write out this integer X in decimal notation. Clearly, we are satisfied to have obtained the integer in one of its representations and do not attempt to convert one representation into another one. The machine is, of course, also capable of such sophistication, and hence can digest the integer X. In regard to all known finite operations on the integers, the machine has already surpassed us in speed and accuracy.

Hence the machine appears no weaker than our intuition is handling the integers. From Bishop's thesis that all mathematics is derived from the integers, it follows as a corollary that the machine is equipotent as our intuition in its ability to grasp mathematics. Therefore, at least in principle, informal and formal constructive mathematics are equivalent. However, in practice, they are very different. Why is this so? The answer lies not only in the greater complexity of the brain over the internal hardware of the machine, but also in the greater sophistication of human mathematical languages over the current computer languages. For instance, the beautiful Fourier inversion formula

$$(*) \quad f(X) = \frac{1}{2\pi}\int_{-\infty}^{\infty}\left[\int_{-\infty}^{\infty} f(\alpha)e^{i\alpha\omega}d\alpha\right]e^{-i\omega X}\,d\omega$$

defies any exact, not approximate, representation as an ALGOL program. However, we have no doubt that formula (*) will be computerized, perhaps not in the near future. For the moment, it suffices to give an outline of how this can be done. We examine first (*) from the informal viewpoint. In other words, how did we get (*) from our intuition of the integers? We use the natural inductive process to construct a hierarchy of sets, partially ordered in degrees of complexity. The most basic set of constructive mathematics is again the integers Z. Then we construct the rationals Q out of the integers as ordered pairs of integers whose second components are non-zero. Then we construct the reals R as Cauchy sequences of rationals; that is, as certain functions from Z into Q. We push on to obtain the set of complex numbers C, the set of continuous functions on R, the set of absolutely integrable functions on R, and finally the set S of all continuous and absolutely integrable complex functions on R. Now, formula (*) is clearly a formula on elements of S. This ability to define new sets cannot be over-exaggerated. Once real numbers are properly defined and shown to obey certain laws of operation, we can usually forget them as being Cauchy sequences of rationals and work with them as pure abstract objects. Our attention span is quite limited, and to continuously think on more than two levels of the hierarchy of sets can be very distracting (11). Besides, we sometimes deliberately omit the explicit dependence of a set on the integers, as in the case of an abstract group. When we prove a theorem on abstract groups, the dependence of an abstract group on the integers is hypothetical; that is, we will supply the dependence only on demand, when we specialize to a given concrete group. So this is the informal viewpoint.

How do we then get hold of formula (*) from the formal constructive viewpoint? Here the road is much more arduous; but in principle, we merely formalize each of the informal steps taken to obtain (*). We must define a hierarchy of data types for representation on the machine. This is not possible in such languages as ALGOL 60 or FORTRAN. But we already see in ALGOL 68 the provision that does allow the programmer to construct certain data types of his own, with the generality that routines themselves may be elements of a given type and therefore can be used as parameters in other routines (12). This scheme also allows the construction of types, having, for instance, abstract groups as their elements. One drawback, though, is the a priori definition of the type 'real', without reference to the type 'integer'. This is not surprising because ALGOL is very much an applications-oriented language.

So there is no theoretical difficulty in creating a computer language to accommodate the algorithms of formal constructive mathematics, since these algorithms are merely elements of higher order data types. Unfortunately (or fortunately, depending on your point of view) this is not the case for the proofs in formal constructive mathematics. (The following exposition is due to Bishop (13).) Consider the general implication formula $S \rightarrow T$, where S and T may or may not contain additional implications. Then we have roughly the following interpretations. Classically, $S \rightarrow T$ means $(\sim S) \vee T$. Since this interpretation uses the principle of the excluded middle, it cannot possibly be implemented on the computer. Intuitionistically, Brouwer said that $S \rightarrow T$ means the truth of S necessarily entails the truth of T. This is still too nebulous for the computer. From the informal constructive viewpoint, Bishop says $S \rightarrow T$ means there exists a method (dependent on S and T) which converts a proof of S into a proof of T. This is much closer to an explication of our intuitive (numerical) notion of implication. However, we have no doubt that the most precise and revealing interpretation of implication shall come from formal constructive mathematics. We hope to prescribe a general (finite) algorithm (independent of S and T) that will mechanically transform an algorithm asserting S into an algorithm asserting T. Consider the formula (*) again. Let S be the statement "f is continuous and absolutely integrable on R." Then we have the theorem S $\to$ ($\star$). Then formally, we expect the existence of a general algorithm that will mechanically transform any algorithm which asserts S into an algorithm which asserts ($\star$). We are still searching for this general algorithm. More likely, we will not find a single general algorithm, but a hierarchy of algorithms that transform other algorithms, like a compiler. This hierarchy may look very similar to our hierarchy of data types, partially ordered in complexity.

In conclusion, we see that constructive mathematics serves as a most natural impetus for the search for more powerful computer languages. In turn, computer languages will give us insights in the objects (or data types) and the proof procedures of constructive mathematics.

REFERENCES:

(1) van der Waerden, B. L., Science Awakening, John Wiley & Sons, Inc., New York, 1963

(2) Hardy, G. H., A Mathematician's Apology, Cambridge University Press, London, 1941

(3) Brouwer, L. E. J., "On the Significance of the Principle of Excluded Middle in Mathematics, Especially in Function Theory," in Jean van Heigenoort's book From Frege to Gödel, Harvard University Press, Cambridge, 1967

(4) Kleene, S. C., Introduction to Metamathematics, D. van Nostrand Co., Inc., Princeton, 1952

(5) Bishop, E., "Mathematics as a numerical language" in Intuitionism and Proof Theory, edited by Kino, Myhill and Vesley, North-Holland Pub. Co., Amsterdam, 1970

(6) Bishop, E., Foundations of Constructive Analysis, McGraw-Hill, New York, 1967

(7) Bishop, E. and Cheng, H., Constructive Measure Theory, Memoir 116 of American Mathematical Society, Providence, 1972

(8) Chan, Y. K., "A constructive approach to the theory of stochastic processes," Trans. of AMS, Vol. 165, 1972, p. 37

(9) Nuber, J., "A Constructive Ergodic Theorem," to appear

(10) Bishop, E., "How to Compile Mathematics into ALGOL," to appear

(11) Strachey, C., "System Analysis and Programming" in Information, A Scientific American Book, W. H. Freeman & Co., San Francisco, 1966

(12) Branquart, P., Lewi, J., Sintzoff, M., and Wodon, P. L., "The Composition of Semantics in ALGOL 68," Communications of ACM, Vol. 14, 1971, p. 697

(13) Bishop, E., "A Formal Language for Constructive Mathematics," to appear