THE BULLETIN OF SYMBOLIC LOGIC
Volume 2, Number 3, Sept. 1996

COMPUTABILITY AND RECURSION

ROBERT I. SOARE

Abstract. We consider the informal concept of "computability" or "effective calculability" and two of the formalisms commonly used to define it, "(*Turing*) *computability*" and "(*general*) *recursiveness*". We consider their origin, exact technical definition, concepts, history, general English meanings, how they became fixed in their present roles, how they were first and are now used, their impact on nonspecialists, how their use will affect the future content of the subject of computability theory, and its connection to other related areas.[1]

After a careful historical and conceptual analysis of computability and recursion we make several recommendations in section §7 about preserving the *intensional* differences between the concepts of "computability" and "recursion." Specifically we recommend that: the term "recursive" should no longer carry the additional meaning of "computable" or "decidable;" functions defined using Turing machines, register machines, or their variants should be called "computable" rather than "recursive;" we should distinguish the intensional difference between Church's Thesis and Turing's Thesis, and use the latter particularly in dealing with mechanistic questions; the name of the subject should be "*Computability Theory*" or simply *Computability* rather than "Recursive Function Theory."

§1. Introduction. All functions are on the nonnegative integers, $\omega = \{0, 1, 2, \dots\}$, and all sets will be subsets of ω. The central concept of the field of computability theory is the notion of an "effectively calculable" or "computable" function.

DEFINITION 1.1. A function is "*computable*" (also called "*effectively calculable*" or simply "*calculable*") if it can be calculated by a finite mechanical procedure. (For a more precise description see §3.1.)

Received March 22, 1996, and in revised form, June 18, 1996.

The author was partially supported by National Science Foundation Grant DMS 94-00825. Some of this material was given in the first part of a lecture at the 10th International Congress for Logic, Methodology and Philosophy of Science, Section 3: Recursion Theory and Constructivism, August 19–25, 1995, which will appear in Soare [ta2].

[1]Helpful suggestions, comments, and criticisms on preliminary drafts of this paper were made by: W. Calhoun, P. Cholak, B. Cooper, M. Davis, R. Downey, S. Feferman, L. Fortnow, R. Gandy, W. Gasarch, L. Harrington, C. G. Jockusch, A. H. Lachlan, S. Lempp, M. Lerman, S. MacLane, G. Mints, J. Owings, H. Putnam, G. E. Sacks, M. Schaefer, J. Shoenfield, R. A. Shore, W. Sieg, T. Slaman, M. Stob, G. Tamburrini, D. van Dalen, and S. S. Wainer, among others. However, any errors are entirely the author's responsibility.

1079-8986/96/0203-0002/$4.80

DEFINITION 1.2. (i) A function is *Turing computable* if it is definable by a Turing machine, as defined by Turing *1936*. (See [Kleene, *1952*] or [Soare, *1987*].)

(ii) A set A is *computably enumerable* (*c.e.*) if A is $\emptyset$ or is the range of a Turing computable function.

(iii) A function f is *recursive* if it is *general recursive*, as defined by Gödel *1934*. (See also Kleene's variant *1936*, *1943*, and [*1952*, p. 274].)

(iv) A set A is *recursively enumerable* (*r.e.*) if A is $\emptyset$ or is the range of a general recursive function.[2]

Later we shall say more of these *formal* definitions and their meanings. For the moment we regard these terms strictly with their *intensional* meaning as above, and we do not *extensionally* identify them with each other or with other formal notions known to be mathematically equivalent (such as λ-definability, μ-recursiveness, or Post normal systems described later), nor with the *informal* notions of computable or effectively calculable under Church's Thesis or Turing's Thesis.

Several logicians have remarked that it is curious that the subject of computability theory (hereinafter called "*the subject*") is called "*recursive function theory*" or "*recursion theory*" instead of the seemingly more natural, "*computability theory,*" as we shall call it here. For example, Feferman *1992* [§6.6] wrote, "... the mathematical theory of computability—in the guise of recursion theory." For several decades there has been a gradual tendency to replace the name "recursion theory" by "computability theory" especially in titles or subtitles of books, for example, Cutland *1980* or the subtitle of Soare *1987*, *A Study of Computable Functions and Computably Generated Sets*.

The purpose of this paper is to examine the meaning, origin, and history of the concepts "recursive" and "computable" with an eye toward re-examining how we use them in practice. The ultimate aim is to ask: "*What is the subject really about?*" For example, is it about computability, recursions, definability, or something else?

In §2 we review the origin and history of each concept and the formal definitions. In §3 we consider the Church-Turing Thesis that the intuitively computable functions coincide with the formally computable ones, and consider using the thesis as a definition. In §4 we trace the historical development of certain parts of the subject after the 1930's and show how the present practices were adopted. General English usage is discussed in

[2]This terminology is the same as that introduced in the 1930's and used since then, except for the term "computably enumerable," recently introduced, because Turing and Gödel did not explicitly introduce a term for these corresponding *sets*, but just for the computable *functions*. Post *1944* explicitly added the empty set as an r.e. set (see [Davis, *1965*, p. 308]), which Church and Kleene had omitted.

§5, and the themes and goals of the subject in §6. The next section, §7, contains an analysis and recommendations based on the rest of the paper, and §8 contains some concluding remarks and a quote by Gödel about the absoluteness of computability.

We cite references by their number in the order listed in our bibliography but also in the usual convention by author and year, *e.g.*, [Post, *1944*] or simply Post *1944*, with the year in italics. To save space we omit from our bibliography some references which appear in Soare *1987*, van Heijenoort *1967*, or Kleene *1952*, and we cite them as there by year.

§2. A brief history of computability.

2.1. The concepts of computability and recursion. A *computation* is a process whereby we proceed from initially given objects, called *inputs*, according to a fixed set of rules, called a *program*, *procedure*, or *algorithm*, through a series of *steps* and arrive at the end of these steps with a final result, called the *output*. The algorithm, as a set of rules proceeding from inputs to output, must be precise and definite, with each successive step clearly determined. The concept of *computability* concerns those objects which may be specified in principle by computations, and includes relative computability (computability from an oracle as explained in §4.3) which studies the relationship between two objects which holds when one is computable relative to the other. For the Gödel-Church-Turing case of computability on ω (called ω-*computability theory*) the inputs, outputs, the program, and computation will all be finite objects, but in Kleene's higher order computability such as computability on constructive ordinals, or higher types, computations may be more general objects such as finite path trees (well-founded trees), and the inputs may be infinite objects such as type 1 objects, namely functions from ω to ω. In α-computability theory (computability on admissible ordinals) the inputs and outputs are likewise suitable generalized.

The concept of *recursion* stems from the verb "recur," "to return to a place or status." (See §5.) The primary mathematical meaning of recursion (§5) has always been "definition by induction" (*i.e.*, by recursion), namely defining a function f at an argument x using its own previously defined values (say $f(y)$ for $y < x$), and also using "simpler" functions g (usually previously defined). The advantage of the Herbrand-Gödel definition of a (general) recursive function (§2.4) was that it encompassed recursion on an arbitrary number of arguments, and many felt it "included all possible recursions."

The Kleene Fixed Point Theorem gave a still more powerful form of this "reflexive program call" permissible in programs. Let $\{P_n\}_{n\in\omega}$ be an effective listing of all (Turing) programs and let φ_n be the computable partial function computed by P_n. The Kleene Fixed Point Theorem (Recursion

Theorem) asserts that for every Turing computable total function $f(x)$ there is a fixed point n such that $\varphi_{f(n)} = \varphi_n$. This gives the following recursive call as described in [Soare, *1987*, pp. 36–38]. Using the Kleene s-m-n-theorem we can define a computable function $f(x)$ by specifying $\varphi_{f(x)} : \ldots x \ldots$, according to some program $P_{f(x)}$, which may mention x and may even call program P_x. Taking a fixed point n for $f(x)$ we have $\varphi_n = \varphi_{f(n)} : \ldots n \ldots$, so that the program P_n for computing φ_n can in effect "call itself" (or more precisely call a program which computes the same function) during the execution of the program. We call this a "*reflexive program call*". Platek's thesis *1966* in higher types stresses the role of fixed points of certain functionals and is often cited as an example of a more general type of recursion. (See §4.4.)

The *concept of recursion* used here includes: (1) induction and the notion of *reflexive program call*, (including primitive recursion and also Kleene's Recursion Theorem); (2) the notion of a *fixed point* for some function, and the more general Platek style fixed points in higher types (see §4.4); (3) other phenomena related to (1) and (2) specified for certain situations and structures. However, the concept of recursion does *not* include the notion of "computable" or "algorithmic" as described in the first paragraph.

2.2. The origin of recursion. Well before the nineteenth century mathematicians used the principle of defining a function by induction. Dedekind *1888* proved, using accepted axioms, that such a definition defines a *unique* function, and he applied it to the definition of the functions $m + n$, $m \times n$, and m^n. Based on this work of Dedekind, Peano *1889* and *1891* wrote the familiar five axioms for the positive integers. As a companion to his fifth axiom, mathematical induction, Peano used definition by induction, which has been called *primitive* recursion (since Péter *1934* and Kleene *1936*), namely

$$(1)\qquad \text{Scheme (V)} \qquad \begin{cases} f(0, \overrightarrow{y}) = h(\overrightarrow{y}) \\ f(x+1, \overrightarrow{y}) = g(x, f(x, \overrightarrow{y}), \overrightarrow{y}) \end{cases}$$

where g and h are previously defined functions, and $\overrightarrow{y}$ denotes a (possibly empty) sequence, $y_1, \ldots, y_n$, of additional variables (parameters). This is Scheme (V) in the well-known five schemata used to define the class of primitive recursive functions, see Soare [*1987*, pp. 8–9], or Kleene, [*1952*, p. 219]. The other schemata are: (I) successor $\lambda x\,[x + 1]$; (II) constant functions $\lambda x_1 \ldots x_n\,[k]$; (III) projections $\lambda x_1 \ldots x_n\,[x_i]$; and (IV) composition $f(\overrightarrow{x}) = h(g_1(\overrightarrow{x}), \ldots, g_m(\overrightarrow{x}))$. The concept of recursion played an important role in the foundations of mathematics and in the work of Skolem *1923*, Hilbert *1926*, Gödel *1931*, and Péter *1934*.

2.3. The origin of computable functions. Mathematicians have studied calculation and algorithms since the time of the Babylonians. Kleene [Kleene, *1988*, p. 19] wrote, "The recognition of algorithms goes back at least to Euclid (c. 330 B.C.)." For example there is Euclid's famous greatest common divisor algorithm. The name "algorithm" comes from the name of the ninth century Arabian mathematician Al-Khowarizmi. Along with the development of theoretical mathematical algorithms there developed an interest in actual calculating machines. In 1642 the French mathematician and scientist, Blaise Pascal, invented an adding machine which may be the first digital calculator. In 1671 the German mathematician and philosopher, Gottfried Wilhelm Leibniz, co-inventor with Newton of the calculus, invented a machine that performed multiplication. Leibniz's machine, called a *stepped reckoner* could not only add and multiply, it could divide, and extract square roots, by a series of repeated additions, used even today. His stepped gear wheel still appears in a few twentieth century devices. Leibniz' main contribution was the demonstration of the superiority of the binary over the decimal representation for mechanical computers. In the work of Leibniz the symbolic representation of problems was combined with a search for their algorithmic solutions. Sieg [Sieg, *1994*, p. 73] wrote that Leibniz "viewed algorithmic solutions of mathematical and logic problems as paradigms of problem solving in general. Remember that he recommended to disputants in *any* field to sit down at a table, take pens in their hands, and say 'Calculemus'!" Leibniz searched for a universal language (*lingua characteristica*) and a calculus of reasoning ("*calculus ratiocinator*") with which to facilitate his program. Around 1834 Babbage invented the idea of an "Analytic Engine," which could be programmed to perform long and tedious calculations, and formulated what Gandy [*1988*, p. 58] called "Babbage's Thesis," that "the whole of the development and operations of analysis are now capable of being executed by machinery." Gandy [*1988*, p. 57] pointed out that considering Babbage's Analytic Engine as a register machine, his proposed operations define precisely the Turing computable functions.

2.4. General recursive functions. These two trends of recursion and computability were brought together in the 1930's by Gödel, Church, Kleene, Turing, and others partly in response to questions raised earlier by Hilbert. At the end of the nineteenth century Hilbert *1899* gave an axiomatization of geometry and showed *1900* that the question of the consistency of geometry reduced to that for the real-number system, and that in turn to arithmetic by results of Dedekind (at least in a second order system). Hilbert *1904* proposed proving the consistency of arithmetic by what became known by *1928* as his *finitist program*. He proposed using the finiteness of mathematical proofs in order to establish that contradictions could not be derived. This tended to reduce proofs to manipulation of finite strings of symbols devoid of intuitive meaning which stimulated the development of mechanical processes to accomplish this. Closely related was the *Entscheidungsproblem*,[3] the decision problem for first order logic, which emerged in the early 1920's in lectures by Hilbert and was described in Hilbert and Ackermann *1928*. It was to give a decision procedure [*Entscheidungsverfahren*] "that allows one to decide the validity[4] (respectively satisfiability) of a given logical expression by a finite number of operations" (Hilbert and Ackermann [*1928*, p. 72–73]). Hilbert characterized this as the fundamental problem of mathematical logic.

Gödel *1931* proved his *first* incompleteness theorem which (stated in modern terms and with an improvement by Rosser) asserts roughly that any consistent extension T of elementary number theory is incomplete. By arithmetizing the proof Gödel obtained his *second* incompleteness theorem which asserts that such a T cannot prove its own consistency (see [Kleene, *1952*, §42]), which was a setback for Hilbert's program (see Gödel *1958*).

In his proof Gödel *1931* [Gödel, *1986*, p. 158] used the notion of a primitive recursive function (which he called "recursive" [*eine rekursive Funktion*]) because these functions were easily representable in Gödel's formal system **P** for arithmetic, and were sufficient to enable him to "Gödel number" all the syntactic objects so that he could obtain self-reference and thereby incompleteness. Gödel realized, however, that the primitive recursive functions did not include *all* effectively calculable functions,[5] and in *1934* he proposed a wider class of functions based on an earlier suggestion of Herbrand. Gödel called these the *general recursive functions*. Herbrand had written Gödel a letter on April 7, 1931 (see Gödel [*1986*, p. 368] and Sieg [*1994*, p. 81]), in which he wrote, "If φ denotes an unknown function, and $\psi_1, \ldots, \psi_k$ are known functions, and if the ψ's and φ are substituted in one another in the most general fashions and certain pairs of resulting expressions are equated, then if the resulting set of functional equations has one and only one solution for φ, φ is a recursive function." Gödel made two restrictions on this definition to make it *effective*, first that the left-hand sides of the functional equations be in standard form with φ being the outermost symbol, and second that for each set of natural numbers $n_1, \ldots n_j$ there exists a unique m such that $\varphi(n_1, \ldots n_j) = m$ is a derived equation. Kleene *1936*, *1943*, and *1952* introduced variants of Gödel's two rules which give an equivalent formulation of the Herbrand-Gödel definition.

2.5. The flaw in Church's Thesis. In 1930 Church had been studying a class of effectively calculable functions called *λ-definable functions*. Church's

[3]Kleene [*1987b*, p. 46] states, "The *Entscheidungsproblem* for various formal systems had been posed by Schröder *1895*, Löwenheim *1915*, and Hilbert *1918*."

[4]Here "valid" means "true in the standard structure," not the modern sense of valid as true in all structures.

[5]Ackermann in 1928 had produced a function defined by double recursion which was not primitive recursive.

student, Kleene, showed by 1933 that a large class of number theoretic functions were λ-definable. On the strength of this evidence, Church proposed to Gödel around March, 1934 [Davis, *1965*, pp. 8–9] that the notion of "effectively calculable" be identified with "λ-definable," a suggestion which Gödel rejected as "thoroughly unsatisfactory."

Following this encounter with Gödel, Church changed formal definitions from "λ-definable" to "recursive," his abbreviation for Herbrand-Gödel general recursive, and Church presented on April 19, 1935, to the American Mathematical Society his famous proposition published in *1936* and known (since Kleene *1952*) as *Church's Thesis* which asserts that the effectively calculable functions should be identified with the recursive functions. This is apparently the first published appearance of the term "recursive" to mean "general recursive." On the basis of this Thesis, Church *1936* announced the unsolvability of Hilbert's *Entscheidungsproblem*.

Gödel, however, remained unconvinced of the validity of Church's Thesis through its publication *1936*. This is all the more significant, first, because Gödel had originated the first formalism, that of the general recursive functions, and the one upon which Church based his Thesis. Second, much of the evidence for Church's Thesis rested on the coincidence of these formal classes, and this was based largely on Kleene's use of arithmetization, the method that Gödel *himself* had introduced so dramatically in *1931*. The reasons why Gödel did not accept or invent the thesis himself are explained in Davis *1982* and below.

The flaw in Church's argument [*1936*, §7] for his thesis was this. Church began by defining an "effectively calculable" function to be one for which "there exists an algorithm for the calculation of its values." Church analyzed the informal notion of the calculation of a value $f(n) = m$ according to a step-by-step approach (so called by Gandy [*1988*, p. 77]) from two points of view, first by an application of an algorithm, and second as the derivation in some formal system, because, as he pointed out, Gödel had shown that the steps in his formal system P were primitive recursive. Following Davis [*1958*, p. 64] or Shoenfield [*1967*, pp. 120–121] it is reasonable to suppose that the calculation of f proceeds by writing expressions on a sheet of paper, and that the expressions have been given code numbers, $c_0, c_1, \dots c_n$. Define $\langle c_0, c_1, \dots c_n\rangle = p_0^{c_0} \cdot p_1^{c_1} \dots p_n^{c_n}$, where p_n denotes the n^{th} prime number. We say that the calculation is *stepwise recursive* if there is a partial recursive function ψ such that $\psi(\langle c_0, \dots, c_i\rangle) = c_{i+1}$ for all i, $0 \leq i < n$.

If the basic steps are stepwise recursive, then it follows easily by the Kleene Normal Form Theorem (see §4.1) which Kleene had proved and communicated to Gödel before November, 1935 (see Kleene [*1987b*, p. 57]), that the entire process is recursive. The fatal weakness in Church's argument was the core assumption that the atomic steps were stepwise recursive, something he did not justify. Gandy [*1988*, p. 79] and especially Sieg [*1994*, pp. 80, 87] in their excellent analyses brought out this weakness in Church's argument. Sieg [p. 80] wrote, "... this core does not provide a convincing analysis: steps taken in a calculus must be of a restricted character and they are assumed, for example by Church, without argument to be recursive." Sieg [p. 78] wrote, "It is precisely here that we encounter the major stumbling block for Church's analysis, and that stumbling block was quite clearly seen by Church," who wrote that without this assumption it is difficult to see how the notion of a system of logic can be given any exact meaning at all. It is exactly this stumbling block which Turing overcame by a totally new approach.

§**3. Turing's contributions to computability.** In the spring of 1935 a twenty-two year old student at Cambridge University, who had just given an independent proof of the Central Limit Theorem [Zabell, *1995*], heard the lectures of Professor M. H. A. Newman on Gödel's paper and on the Hilbert *Entscheidungsproblem*. Turing worked on the problem for the remainder of 1935 and submitted his solution to the incredulous Newman on April 15, 1936. Turing's monumental paper *1936* was distinguished because: (1) Turing analyzed an idealized *human* computing agent (a "*computor*") which brought together the intuitive conceptions of a "function produced by a mechanical procedure" which had been evolving for more than two millenia from Euclid to Leibniz to Babbage and Hilbert; (2) Turing specified a remarkably simple formal device (*Turing machine*) and proved the equivalence of (1) and (2); (3) Turing proved the unsolvability of Hilbert's *Entscheidungsproblem* which established mathematicians had been studying intently for some time; (4) Turing proposed a *universal* Turing machine, one which carried within it the capacity to duplicate any other, an idea which was later to have great impact on the development of high speed digital computers and considerable theoretical importance. Gödel enthusiastically accepted Turing's Thesis and his analysis, and thereafter Gödel always gave credit to Turing (not to Church or to himself) for the definition of mechanical computability and computable function.

3.1. Turing's idealized human computor. In 1935 Turing and everyone else used the term "*computer*" for an idealized *human* calculating with extra material such as pencil and paper, or a desk calculator, a meaning very different from the use of the word today. (Even ten years later in his 1946 report [Turing, *1986*, p. 20] on the Automatic Computing Engine (A.C.E.) Turing used the term "computer" to refer to a human with paper, as in [p. 106]. Turing wrote that A.C.E. can do any job of a (human) computer in one ten-thousandth of the time.) To avoid confusion we shall follow Gandy *1988* and Sieg *1994* and use the term "*computor*" to mean such an *idealized* *human* calculating in a purely mechanical fashion, and the term "*computer*" for a *machine*, either an idealized machine like a Turing machine or register machine, or for a physical device like a high speed digital computer. The analysis in this subsection was not completely clear in Turing *1936*, and is due almost entirely to Sieg *1994* and *1995*, who built upon Gandy *1988*.

To analyze what it means for a function to be "calculable by an algorithm" or a "mechanical procedure," Turing put certain conditions on the calculation. Turing [*1939*, §9, p. 249–254] assumed that the computation was being done by the computor "writing certain symbols on paper," and that the paper was one dimensional and divided into squares. He also proposed a set of *states* (of mind) of the computor. First, Turing required three *finiteness* conditions: (F1) the number of symbols; (F2) the number of squares observed at any one moment; (F3) the number of states. Turing proposed a number of simple operations "so elementary that it is not easy to imagine them further subdivided." Turing allowed the computor to observe a set of squares and in one atomic operation to: change the set of squares being observed or print on an observed square; and change its state in accordance with the following *neighborhood* conditions: (C1) the computor can change the symbol only in an *observed* square and then at most *one* symbol; (C2) the computor can move to a different set of observed squares but only within a certain bounded distance L of an observed square; (C3) the atomic operation must depend only on the current state and the symbols in the observed squares. Turing also imposed a determinacy condition (D) that from the state and observed symbols there was at most one atomic operation which could be performed, but this is unnecessary, since it is now well-known how to simulate a nondeterministic process by a deterministic one.[6]

From the precise description of his computor, Turing then formally defined his familiar *automatic machine*, now known as a *Turing machine*, a finite state machine with a two-way infinite tape, whose squares contained symbols from a finite alphabet, with a read/write head which scans one square at a time, and a finite set of instructions (Turing program), see Soare [*1987*, p. 12]. Turing called a function defined by a Turing machine a "*computable function.*" Using Turing's analysis we can now repair the weakness in Church's argument. (See Sieg [*1994*, p. 95].) To show that an effectively calculable function is recursive, take the algorithm which calculates it, find the technical description of the corresponding computor, then the associated Turing machine, and then the associated recursive function, from the equivalence of the latter two classes.

[6]Hodges [*1983*, p. 96] suggests that Turing's computor may have grown out of his analysis of a typewriter: "Alan had dreamt of inventing typewriters as a boy; Mrs. Turing had a typewriter; and he could well have begun by asking himself what was meant by calling a typewriter 'mechanical.' " Turing's computor does resemble a kind of erasing typewriter with an infinite carriage but with a finite program.

DEFINITION 3.1. A function is *computorable* if it can be calculated by an idealized human computor as defined above.

Turing then proved *Turing's Theorem*: *Any computorable function is Turing computable.* Although not proved in a formal system, Turing's proof is as rigorous as many in mathematics. Gandy [*1988*, p. 82] observed, "Turing's analysis does much more than provide an argument for" Turing's Thesis, "*it proves a theorem.*"[7] Furthermore, as Gandy [*1988*, pp. 83–84] pointed out, "Turing's analysis makes no reference whatsoever to calculating machines. Turing machines appear as a result, a codification, of his analysis of calculations by humans." *Turing's Thesis* [*1936*, §9] is that every intuitively computable function is computable by a Turing machine. By Turing's Theorem, Turing's Thesis reduces to the following thesis (called by Sieg *1994* Turing's "Central Thesis").

> **Turing's Thesis (TT-Computor).**[8] If a function is informally computable (*i.e.*, definable by a finite mechanical procedure or algorithm) then it is *computorable* (*i.e.*, computed by a Turing idealized human computor).

Thus, the relationship between: (1) effectively calculable; (2) computorable; and (3) Turing computable is that: (1) $\implies$ (2) by TT-Computor, and also (2) $\implies$ (3) by Turing's Theorem, as Sieg *1994* and *1995* has also pointed out. Subsequent work has been done to show that more functions fall into one of these two classes. For example, Sieg and Byrnes *1995* generalized the concept of computorable and thus weakened TT-computor and strengthened Turing's Theorem. Similarly, Gandy *1980* analyzed discrete deterministic mechanical devices (DDMD machines) proving them to be Turing computable, a variant of TT known as TT-DDMD. However, these results and other subsequent work do not affect the original Turing Thesis TT-Computor which we regard not so much as a thesis but rather as a *definition* of the two thousand year old notion of an algorithmic function. It is now seen to encompass all modern high speed digital computers as well.

3.2. Accepting Turing's Thesis. If we review the conceptions of algorithms and mechanical procedures over the last two millenia from the Euclidean algorithm, Pascal's and Leibniz' conceptions of calculating, Babbage's analytic

[7] Gandy actually wrote "Church's Thesis" not "Turing's Thesis" as written here, but surely Gandy meant the latter, at least intensionally, because Turing did not prove anything in *1936* or anywhere else about general recursive functions.

[8] Here we follow Gandy *1980* in using appended words or letters to pinpoint the exact version of Turing's Thesis proposed. For example, Gandy [*1980*, p. 124] wrote of "*Theorem T.* What can be calculated by an abstract human being working in a routine way is computable," and distinguishes it from, "*Thesis M.* What can be calculated by a machine is computable." Gandy goes on to propose his own Thesis P about discrete deterministic mechanical devices (DDMD).

engine, Hilbert's and Gödel's computation of a function in a formal system, and many others included in the concept of computation described in §2.1, we see that they all fit within the computor model. Indeed, we claim that the common conception of mechanical procedure and algorithm envisioned over this period is exactly what Turing's computor captures.

This may be viewed as roughly analogous to Euclidean geometry or Newtonian physics capturing a large part of everyday geometry or physics, but not necessarily all *conceivable* parts. Here, Turing has captured the notion of a function computable by a mechanical procedure, and as yet there is no evidence for any kind of computability which is *not* included under this concept. If it existed, such evidence would not affect Turing's Thesis about mechanical computability any more than hyperbolic geometry or Einsteinian physics refutes the laws of Euclidean geometry or Newtonian physics. Each simply describes a different part of the universe.

Turing machines and Turing's analysis were enthusiastically accepted by the founders of the subject, Gödel, Church, and Kleene, as the correct definition of computability. In [*193?*][9] [Gödel, *1995*, p. 168] Gödel wrote regarding the formal definitions of computability, "That this really is the correct definition of mechanical computability was established beyond any doubt by Turing." Gödel left no doubt that he regarded Turing's approach as superior to all other previous definitions (including his own recursive functions) when he wrote in *1964* [Davis, *1965*, p. 72, footnote], speaking of Turing machines, that, "As for previous equivalent definitions of computability, which, however, are much less suitable for our purpose, see A. Church *1936*, pp. 256–358." (Gödel's reference is to Church's Thesis §9 which we have just analyzed in §2.5.) Kleene wrote [*1981b*, p. 49], "Turing's computability is intrinsically persuasive" but "λ-definability is not intrinsically persuasive" and "general recursiveness scarcely so (its author Gödel being at the time not at all persuaded)." Church, in his review *1937* of Turing *1936*, wrote that of the three different notions: computability by a Turing machine, general recursiveness of Herbrand-Gödel-Kleene, and λ-definability, "The first has the advantage of making the identification with effectiveness in the ordinary (not explicitly defined) sense evident immediately—*i.e.*, without the necessity of proving preliminary theorems." Most people today accept Turing's Thesis. Sieg [*1994*, p. 96] wrote, "Thus, Turing's clarification of effective calculability as *calculability by a mechanical computor* should be accepted."

Some have cast doubt on Turing's Thesis on the grounds that there might be physical or biological processes which may produce, say, the characteristic

[9]In referencing this paper [*193?*] of Gödel we follow the bibliographic referencing and numbering in his collected papers [Gödel, *1995*, p. 156] where the editors use "[*193?*]" and explain, "This article is taken from handwritten notes in English, evidently for a lecture, found in the *Nachlass* in a spiral notebook. Although the date of the piece is not known, some conjectures about this will be discussed later."

function of the halting problem. It is possible that these may exist (although there is presently no evidence) but if so, this will have absolutely *no effect* on Turing's Thesis because they will not be algorithmic or mechanical procedures as required in §2.1 and in Turing's Thesis. Although suggesting the possibility of noncomputational *mental* processes [Gödel, *1995*, p. 310], Gödel was unequivocal in his support of Turing's Thesis TT-Computor. Regarding the possibility of other nonmechanical procedures, Gödel *1964* wrote,

> Note that the question of whether there exist *non-mechanical* procedures not equivalent with any algorithm, has nothing whatsoever to do with the adequacy of the definition of "formal system" and of "mechanical procedure."
> —*Gödel* 1964 [Davis, *1965*, p. 72].

3.3. The Church-Turing Thesis as a definition. When Church *1936* first proposed Church's Thesis, he thought of it as a *definition*, not as a *thesis*. Church [Davis, *1965*, p. 90] wrote, "The purpose of the present paper is to propose a definition of effective calculability." Similarly, Turing *1936* did not use the term "definition," but he spoke [Davis, *1965*, p. 135] of showing "that all computable numbers are [Turing] 'computable', " and clearly regarded it as the definition of computable. Gödel stated on several occasions that the correct *definition* of computability had unquestionably been achieved by Turing.

> ... one has for the first time succeeded in giving an absolute definition of an interesting epistemological notion, *i.e.*, one not depending on the formalism chosen. ... For the concept of computability, however, although it is merely a special kind of demonstrability or decidability, the situation is different. By a kind of miracle it is not necessary to distinguish orders, and the diagonal procedure does not lead outside the defined notion.
> —*Gödel: 1946 Princeton Bicentennial,* [Gödel, *1946*, p. 84].

> The greatest improvement was made possible through the precise definition of the concept of finite procedure, ... This concept, ... is equivalent to the concept of a "computable function of integers" ... The most satisfactory way, in my opinion, is that of reducing the concept of finite procedure to that of a machine with a finite number of parts, as has been done by the British mathematician Turing.
> —*Gödel: Gibbs lecture* 1951 [Gödel, *1995*, pp. 304–305].

> But I was completely convinced only by Turing's paper.
> —*Gödel: letter to Kreisel of May 1, 1968* [Sieg, *1994*, p. 88].

The theses of Church and Turing were not even called "theses" at all until Kleene [*1943*, p. 60] referred to Church's "definition" as "Thesis I," and then in *1952* Kleene referred to "Church's Thesis" and "Turing's Thesis." What is even more curious is that the phrase "Church's Thesis" came to denote also "Turing's Thesis" and perhaps others as well, thereby blurring all intensional distinctions. (This, of course, stems partly from the Recursion Convention in §4.6 that "recursive" denotes "computable," because under this convention Turing's Thesis follows from Church's Thesis, whereas in reality the reverse was true as seen in §3.1.) There are many examples of this in the literature. For example, Gandy's *1980* paper is entitled, "*Church's thesis and principles for mechanisms*". However, Gandy's paper is entirely about *Turing's* Thesis and whether or not certain intuitively defined classes are *Turing* computable (*i.e.*, *mechanistic*), not whether or not they are *recursive*. The hypotheses are stated in terms of variants of Turing's Thesis (TT) such TT-H, TT-M, TT-DDMD. They are presented both informally and formally entirely in the language of *machines*. Gandy's main result is that what can be calculated by a discrete deterministic mechanical device (DDMD) is Turing computable.

In contrast, the distinction of *intensional* meaning which distinguishes between Church's Thesis and Turing's Thesis is preserved by others, for example Sieg *1994* and Tamburrini *1995*, and here. Here we also use the phrase "*Church-Turing Thesis (CTT)*" to refer to the amalgamation of the two theses (these and others) where we identify all the informal concepts of Definition 1.1 with one another and we identify all the formal concepts of Definition 1.2, and their mathematical equivalents, with one another and suppress their intensional meanings.

We now propose that Turing's Thesis be used as a *definition* of a computable function as Turing and Gödel suggested. Other theses in the past have dealt with very problematic topics but have eventually become definitions as we now discuss.

3.4. Other theses became definitions. One senior logician objected to this proposed definition because he said we should view the Church-Turing Thesis as certainly correct, but as "a one of a kind, without any true analogue in mathematics. I think we recursion theorists should be proud of this, and not (as you seem to suggest) replace it by a change of our definitions." There is no reason why we cannot use Turing's Thesis as a definition of computability and still maintain awe and pride at a fundamental discovery. As to the uniqueness of this discovery in the history of mathematics, it is informative to consider the history of other "theses." In the early 1800's mathematicians were trying to make precise the intuitive notion of a continuous function, namely one with no breaks. What we might call the "Cauchy-Weierstrass Thesis" asserts that a function is intuitively continuous iff it satisfies the usual formal δ-ε-definition found in elementary calculus books. Similarly, what we might call the "Curve Thesis" asserts that the intuitive notion of the length of a continuous curve in 2-space is captured by the usual definition as the limit of sums of approximating line segments. The "Area Thesis" asserts that the area of an appropriate continuous surface in 3-space is that given by the usual definition of the limit of the sum of the areas of appropriate approximating rectangles. These are no longer called *theses*, rather they are simply taken as *definitions* of the underlying intuitive concepts.

The same senior logician argued that these analogies are misleading because "Only a moment's thought is needed to see that Weierstrass' definition is a correct formulation of the intuitive notion of continuity. However, it takes a lot of thought to convince oneself that every function which can be computed by an algorithm can be computed by a Turing machine."

This impression of the simplicity in verifying the other theses and the belief in the unique historical place of the Church-Turing Thesis in formally capturing a difficult intuitive notion seems to ignore the history of the other "theses." What is problematic to one generation seems the obvious definition to another. Kline [Kline, *1972*, p. 354] wrote,

> "Up to about 1650 no one believed that the length of a curve could equal exactly the length of a line. In fact, in the second book of *La Geometrie*, Descartes says the relation between curved lines and straight lines is not nor ever can be known. But Robertval found the length of an arch of a cycloid. The architect Christopher Wren (1632–1723) rectified the cycloid ... Fermat, too, calculated some lengths of curves. These men usually found the sum of the segments, then let the number of segments become infinite as each got smaller."

Kline asserts (p. 355) that finding the lengths of curves was one of "the four major problems that motivated the work on the calculus." Regarding the Area Thesis Kline remarked (p. 355) that during the same period Huygens "was the first to give results on the areas of surfaces beyond that of the sphere."

A second distinguished senior logician stated that for him the Curve Thesis is *more* difficult to accept than Turing's Thesis and explained his reasons with references from Kline. With the Curve Thesis there is no upper bound closing downward toward the length of the curve, but merely the *lower* bound of the sum of the lengths of line segments, which increases with ever finer subdivisions. Likewise, for the Area Thesis (unlike the area under a curve in 2-space) there is no *upper* bound to the area, but merely the sum of the areas of finitely many rectangles approaching the correct value from *below*. He notes that *in all three cases,* the length of a curve in 2-space, the area of a surface in 3-space, and the set of all computable functions, *there is no upper bound,* just a lower bound. Furthermore, in both the Curve Thesis and Turing's Thesis one breaks the demonstration into smaller and smaller pieces until it becomes evident.[10]

3.5. Register machines. Closely related to Turing machines is the formalism proposed much later of *register machines* by Shepherdson and Sturgis *1963*. (See also Cutland *1980*, or Shoenfield *1991*.) These have the advantage of more closely resembling modern digital computers which manipulate data and instructions stored in various "registers" rather than having to go back and forth through the data on a single tape. In the version of Cutland [*1980*, p. 9] the register machine contains an infinite number of registers $\{R_n\}_{n\in\omega}$, each of which contains an integer, r_n. The program P is a finite set of instructions built up from the four basic instructions: zero $Z(n)$ (replace r_n by 0), successor $S(n)$ (replace r_n by $r_n + 1$), transfer $T(m, n)$ (replace r_n by r_m), and jump instructions $J(m, n, q)$ (if $r_m = r_n$ go to the qth instruction of P, and otherwise go to the next instruction of P). It is easily shown that Turing machines can compute the same class of functions as register machines.

§4. Later developments in computability.

4.1. Kleene's normal form and his μ-recursive functions. From 1931 to 1934 Kleene tested many operations on functions to see whether they preserve λ-definability. Among these was the least number operator, "the least y such that," which, since [Kleene, *1938*], has been denoted by "μy." Kleene proved that if $R(x, y)$ is a λ-definable relation then so is the partial function $\psi(x) = (\mu y)R(x, y)$.

Kleene used this to prove his Normal Form Theorem *1936* and *1943* which asserts that there is a primitive recursive predicate $T(e, x, y)$ and a primitive recursive function $U(y)$ such that for any general recursive function $\varphi(x)$, there is an index e (corresponding to the system E of equations defining φ) such that

$$\varphi(x) = U(\mu y\ T(e, x, y)). \tag{2}$$

(This is the *1943* version. The *1936* version had a U with an additional parameter.) Kleene's Normal Form Theorem establishes that every general

[10]The second logician pointed out that Kline goes on to say that during the second half of the 17th century various curves were rectified (using essentially the modern definition of arc length). Kline [p. 107] describes how other axioms involve the lengths of concave curves and surfaces. Kline tells how Archimedes deals axiomatically with arc length, and describes how Archimedes gave what can be construed as a proof of the Curve Thesis for certain curves since his axiom gives a way of handling an upper-bound on the length. The second logician suggested that this is analogous to Gandy's proof of TT-DDMD in §3.1, and stated "there is apparently no Gandy-like proof of the Curve Thesis for arbitrary rectilinear curves; in that case arclength is a definition. On the other hand there is a Gandy-like proof of Turing's Thesis for the case of TT-DDMD (namely Gandy's)."

recursive (partial) function is μ-recursive, and conversely. Since the application of μ often leads to only *partial* functions Kleene *1938* introduced the *partial recursive functions*, (*i.e.*, *computable partial functions*). The Normal Form Theorem also holds if we replace total by *partial* recursive functions. Define the class $\mathcal{C}$ to be the smallest class of partial functions closed under the five schemata for primitive recursion (see §2.2) and, in addition, the following schema,

Scheme (VI) (Unbounded Search) $\qquad \varphi(x) = (\mu y)\,[g(x, y) = 0],$

where $g(x, y) \in \mathcal{C}$ and $g(x, y)$ is total. Scheme (VI) is also sometimes called "minimalization," or the "least number operator." Kleene *1952* referred to this $\mathcal{C}$ as the class of (*partial*) μ-*recursive* functions (reserving the term "*recursive*" for Herbrand-Gödel recursive), and used the term (partial) μ-recursive in later papers such as *1959* and *1963*.

The (partial) μ-recursive functions constitute a robust class and one which plays a very important role in the subject. For example, by Gödel numbering the configurations of a computation we can easily prove a normal form theorem for the Turing computable functions (see Soare [*1987*, p. 15]). The μ-recursive functions are a mathematically definable class of functions almost independent of syntax and formalism. They have sometimes been used as the *definition* of a (partial) recursive function (see the table in §4.6), but when used precisely and by Kleene, the formal meaning of "recursive" has been "defined by a Herbrand-Gödel system of equations."

It has sometimes been erroneously written that an advantage of the formalism of recursive functions or μ-recursive functions is that one can precisely write down a proof in either one of the two, but that this would have been infeasible using Turing machines or λ-definable functions. It is true that the latter two are unsuitable for writing proofs, but the former two are not much more suitable. Kleene [*1981*, p. 62] wrote, "Under Herbrand-Gödel general recursiveness and my partial recursiveness adapted from it one works with systems E of equations that can be very unwieldy." The general recursive formalism has almost *never* been used for writing papers, so the writers are probably using "recursive" to refer to "*μ-recursive,*" which was earlier used by Kleene for writing his proofs from *1936* to *1963* and even later, and by some followers like Sacks in his book *1963* on degrees. These expositions were extremely difficult to read (not unlike machine code) and were virtually completely abandoned by the mid 1960's in favor of the style of Rogers' book *1967* which has prevailed in subsequent texts (given in the table in §4.6). This style is to use rigorous proofs but written in the usual informal mathematical style and usually based on the formalism of Turing machines or the closely related register machines to define necessary items such as the number of steps of the computation, the "use function" measuring the number of oracle squares scanned during a computation, and so on.

4.2. Computably enumerable sets and Post. Since they were motivated by formalizing algorithms and possible decision procedures in connection with Hilbert's *Entscheidungsproblem*, the first formalizations of computability were designed to define a computable *function*. However, it had been recognized that effectiveness also occurs with *generating* objects, such as sets of formulas. Church, in his paper *1936* [Davis, *1965*, p. 96] on Church's Thesis, introduced the term "*recursively enumerable set*" for a set which is the range of a recursive function as in Definition 1.2. This is apparently the first appearance of the term "recursively enumerable" in the literature and the first appearance of "recursively" as an adverb meaning "effectively" or "computably."

Church goes on to prove in §6 and §8 various theorems and corollaries about recursively enumerable sets of well-formed formulas. Church also used the term "*effectively enumerable*" for the *informal* concept of a recursively enumerable set but used the latter for *both* the informal and formal concepts.

In the same year Kleene *1936* mentioned [Davis, *1965*, p. 238] a "*recursive enumeration*" and noted that there is no recursive enumeration of Herbrand-Gödel systems of equations which gives only the systems which define the (total) recursive functions. By a "recursive enumeration" Kleene states that he means "a recursive sequence (*i.e.*, the successive values of a recursive function of one variable)." Effectively enumerable or recursively enumerable sets were not mentioned much thereafter until Post's paper *1943* on normal (production) systems which led to generated sets and then his famous *1944* paper which inaugurated the modern study of computably enumerable sets.

In the same year as Turing *1936*, Post *1936* independently of Turing (but not independently of the work by Church and Kleene in Princeton) defined a "*finite combinatory process*" which closely resembles a Turing machine. From this it is often and erroneously written (Kleene [*1987b*, p. 56] and [*1981*, p. 61]) that Post's contribution here was "essentially the same" as Turing's, but in fact it was much less. Post did not attempt to prove that his formalism coincided with any other such as general recursiveness but merely expressed the expectation that this would turn out to be true, while Turing proved the Turing computable functions equivalent to the λ-definable ones. Post gave no hint of a universal Turing machine. Most important, Post gave no analysis as did Turing in §3.1 above of why the intuitively computable functions are computable in his formal system. Post offers only as a "working hypothesis" that his contemplated "wider and wider formulations" are all "logically reducible to formulation 1." Lastly, Post, of course, did not prove the unsolvability of the *Entscheidungsproblem* because at the time Post was not aware of Turing's paper *1936*, and Post believed that Church had settled the *Entscheidungsproblem*. (Post may have been aware of the flaw in Church's

Thesis discussed in §2.5, and perhaps this is why he objected to the use of the term "definition.")

Later, Post *1941* and *1943* introduced a *second* and unrelated formalism called a *production* system and (in a restricted form) a *normal* system, which he explained again in *1944*. Post's (normal) canonical system is a *generational* system, rather than a *computational* system as in general recursive functions or Turing computable functions, and led Post to concentrate on *effectively enumerable sets* rather than computable functions. He showed that every recursively enumerable set is a normal set (one derived in his normal canonical system) and therefore normal sets are formally equivalent to recursively enumerable sets. Post, like Church and Turing, gave a thesis [*1943*, p. 201] but stated in terms of generated sets and production systems, which asserted that "any generated set is a normal set."

Post used the terms "effectively enumerable set" and "generated set" almost interchangeably, particularly for sets of positive integers. Post [*1944*, p. 285] (like Church *1936*) defined a set of positive integers to be *recursively enumerable* if it is the range of a recursive function and then stated, "The corresponding intuitive concept is that of an *effectively enumerable* set of positive integers." Post [*1944*, p. 286] explained his informal concept of a "generated set" of positive integers this way,

> "Suffice it to say that each element of the set is at some time written down, and earmarked as belonging to the set, as a result of predetermined effective processes. It is understood that once an element is placed in the set, it stays there."

Post then [p. 286] restated his thesis from *1943* that "*every generated set of positive integers is recursively enumerable,*" [the italics are Post's] and he remarked that "this may be resolved into the two statements: every generated set is effectively enumerable, every effectively enumerable set of positive integers is recursively enumerable." Post continued, "their converses are immediately seen to be true."

Hence, this amounts to an assertion of the identification (at least extensionally) of the three concepts. Post accepted the Church-Turing Thesis even though he was reluctant to call it a definition, as Church and Turing would have done. Post [*1944*, p. 307, footnote 4] calls attention to Kleene's first use [*1943*, p. 201] of the word "thesis" in this context, but remarks "We still feel that, ultimately, "Law" will best describe the situation," and Post refers to his *1936* where this term was first proposed. This suggests that Post perhaps thought of the thesis as a kind of natural law like the laws of Newtonian physics.

In his famous and very influential paper *1944*, Post continued with the intuitive concepts of "effectively enumerable" and "generated set," which he explains again at some length. The *formalism* Post used was that of his own normal (production) system, *i.e.*, "normal set." He used the term "recursively enumerable set" (Church's term from *1936*) as a name for *both* his *informal* "effectively enumerable set" and for his *formal* version, "normal set." However, the *concept* or formal *definition* of "recursive" does not enter Post's paper at all, only the *terms* "recursive" and "recursively enumerable." Post's use of the term "recursively enumerable" is one of several ambiguities in the subject (ambiguous at least from an *intensional* viewpoint).

In spite of this ambiguity, Post's entrance on the scene was fortunate for recursively enumerable sets and for the entire subject. Previously, the papers in the subject had been written in the very technical formalism of μ-recursive functions (see the last paragraph of §4.1), with little intuition. Recursively enumerable sets had attracted very little attention since their debut in 1936. Post's papers brought excitement, intuitive appeal, and an informal style of proof, much closer to ordinary mathematical proofs, and represented the real birth of the subject of recursively enumerable sets *1943* and *1944* and degrees of unsolvability *1948*. The results and machinery they generated (Post's problem, Friedberg-Muchnik priority method) not only heavily influenced computability on ω but also provided a goal for higher excursions such as meta-recursion theory, α-recursion theory, recursion in higher types, E-recursion theory, and others. These papers of Post stimulated the entire subject for decades, but unfortunately they simultaneously helped to fix the use of the terms "recursive" and "recursively enumerable" to acquire the additional meanings, "computable" and "computably enumerable."

4.3. History of relative computability. The problem of *computability* of a set A *relative* to a set B is that of giving an algorithm for answering every question of the form "Is $x \in A$" by a computation which asks at most finitely many questions of the form "Is $y_1 \in B$?," ... "Is $y_k \in B$?" The first formal definition of relative computability (also called "relative reducibility") was given by Turing [*1939*, §4], in terms of an "*oracle Turing machine.*" This is best visualized as a Turing machine with an extra infinite "oracle tape" on which is written the characteristic function of B (see Soare [*1987*, p. 47]). Other formal definitions were later given by Kleene *1943* and *1952* of a function φ being *general recursive in* a function ψ if the latter is simply added to the equations E defining φ. Post *1948* formulated another definition by modifying his definition *1943* of a canonical (production) system. These three definitions can be proved to be equivalent. (See Kleene *1952*.) Using Turing reducibility (denoted $A \leq_T B$), we say that two sets A and B have the same *information content* or have the same *Turing degree* if $A \leq_T B$ and $B \leq_T A$. Post *1948* introduced this extremely influential concept of Turing degree, also called *degree of unsolvability.* Kleene and Post *1954* laid the foundation for the abstract structure of the degrees, where there has been much research ever since.

It is interesting that all but one of the texts from the table in §4.6 use Turing machines or their variant, register machines, to define $A \leq_{\mathrm{T}} B$, but they apply the term "recursive in" (rather than "computable in") to the result. For example, Shoenfield uses register machines, and his entire apparatus is machine based, as is all his terminology to formulate the definition. He speaks [p. 40] of an "oracle" for a function "F" in the sense of Turing, asking for a value "we have computed" to be used in "the rest of the computation," "the use of an algorithm," the "notion of a program computing a function for this machine," and the "Φ-machine" (oracle machine) being "obtained from the basic machine by adding all F-instructions for all F in Φ." Yet after all this definitional background which heavily uses both the *formalism* and the *concepts* of *machine computation* but *none* of the formalism or concepts of recursion, Shoenfield concludes with the formal definition , "A function is *recursive relative to* Φ if it is computed by some program for the Φ-machine." This is typical of most of these references and is another instance where a *concept* like computability is used to *define* a function, but then a different *name* like "recursion" is assigned afterward even though the concept of recursion is not used in the definition.

> If we replace recursive by computable in results in recursion theory, we often obtain a statement which is evident, or at least more evident than the original result.
> —*Shoenfield* [1995, p. 15].

4.4. Higher order computability. Kleene opened the frontiers of computability on higher type objects in a series of papers first on constructive ordinals and and hierarchies of number-theoretical predicates[11] and later on computability in higher types. Although Kleene calls the functions here by the *word* "recursive" he often used *concepts* of computability to define, explain, and prove theorems about them. For example, in *1955b* Kleene wrote,

> "By *general recursive* functions (predicates) we mean ones whose values can be computed (decided) by ideal computing machines not limited in their space for storing information. A theory of such machines was given by Turing *1936* and in less detail by Post *1936*."

It is on computability on higher types that the concept of recursion comes into one of its more splendid realizations. An object of type 0 is a number; an object of type $n+1$ is a mapping from the set of objects of type n into ω. Thus, an object of type 1 is a *real* (*i.e.*, identified with a function α from ω to ω). A well-known type 2 object is $\mathbf{E}$ where $\mathbf{E}(\alpha) = 0$ if $(\exists x)[\alpha(x) = 0]$, and $\mathbf{E}(\alpha) = 1$ otherwise.

[11] From this work grew later the very beautiful subject of descriptive set theory, although when he began Kleene was unaware of the work in classical descriptive set theory from the early 1900's.

In order to formally define computable functions of higher type, Kleene *1959* used a *schemata-based* definition very much like that for the μ-recursive functions in §4.1. Kleene began by giving [p. 3] *schemata* (S1) to (S8) which closely resemble the previous primitive recursive schemata (I)–(V) of §2.2. After proving various properties about these primitive recursive functions of higher type, Kleene addressed the general recursive case [§3, p. 10]. Kleene began by talking about Turing oracles and "computations being carried out by a preassigned procedure." To obtain the partial recursive functions Kleene added an additional schema (S9) [p. 13] which is a kind of enumeration schema and, together with (S1) to (S8), forms a huge induction. If instead of schema (S9) one adds a schema (S10) which closely resembles the unbounded search schema (VI) of §4.1 then Kleene obtained the "*partial μ-recursive*" functions which are a strictly smaller class than the partial recursive functions (see Kleene §8.4), unlike the ω case where the two classes coincide.

In a later paper Kleene defined his schema (S11): $\varphi(\theta; \overrightarrow{a}) = \psi(\varphi, \theta; \overrightarrow{a})$, and he declared [*1987c*, p. 358], "This schema gives an absolutely general form of recursion." Later, in his Ph.D. dissertation, Platek developed very elegant abstract forms of recursion in higher types. For example, if H is finitary operation with certain properties and

$$F_{n+1} = H(F_n, x)$$

then $F = \bigcup_{n\in\omega} F_n$ is a fixed point, but is not a recursion on any argument. Some people have cited this work by Kleene, Platek, and others to prove that in higher types recursion plays the main role and computability plays very little role if any, but this is not accurate.

Consider Kleene's papers *1959* and *1963* laying the foundations for higher types. Although Kleene used the *name* recursive for his higher type functions, Kleene used the concept of computability to explain them and to carry out his proofs. From the moment Kleene introduced the general recursive case on p. 10 of *1959* he used the *concept* and terminology of computability, including: "computation," "oracle," "preassigned procedure," "mechanical character," and many more. Words like these, particularly "terminating" or "nonterminating" "computations" occur on average several times per page throughout the rest of the article. For example, Kleene showed "how the inductive definition of $\{z\}(\overrightarrow{a}) \simeq w$ provides a computation process," The "stages" of a computation can be arranged in a tree [p. 22], and termination or nontermination of a computation along a certain branch of the tree [p. 32] is crucial to the overall computation. Kleene went on in *1962* and *1962b* to develop what he called "Turing-machine computable functionals of finite types."

Dag Normann is the author of an authoritative text *1980* on the subject of recursion on countable functionals. Normann gave a lecture at Oberwolfach in January, 1996, a main theme of which was that the subject of higher types has much more to do with computability than with recursion.

It is fair to say that the subject of higher types represents a very interesting and beautiful new arena where *both* the concepts of recursion and computability play a key role. Kleene's work and Platek's have raised the pure concept of recursion to new heights with unexpected discoveries of new kinds of fixed points. At the same time motivation and methods have often been those associated with the concept of computability, suitably generalized. Indeed is there any area of recursion theory which has been opened *merely* to study the concepts of self-reference, fixed points, reflexive call and other aspects of recursion alone with *no* intent of studying the effective or computable content of the new area?

4.5. How the terms became fixed. If both Turing and Gödel, the inventors of the two formal definitions and the two names, preferred the terminology "computable" for this class of functions, how did the word "recursive" become preferred for it and for the subject? When Turing's *1939* paper appeared, he had already been recruited by the British government as a cryptanalyst on September 4, 1939 [Hodges, *1983*, p. 161], three days after Britain was plunged into World War II. Turing played the major role [Hodges, *1983*] in 1940 in breaking the German cipher, Enigma. After the war Turing worked on the design of high speed digital computers, first, at the British National Physical Laboratory from 1945 to 1948 and then at the Computing Machine Laboratory in Manchester from 1948 until his death in 1954. Turing wrote a report *1946* [Turing, *1986*] on the design of A.C.E., a high speed digital computer (partly inspired by his universal Turing machine). Gödel moved to set theory and proved his famous results about the consistency of the axiom of choice and the generalized continuum hypothesis which appeared in 1938 and 1939. He returned to computability with his well-known *Dialectica* paper *1958* in which he speaks of "computable functions of finite type" [Gödel, *1990*, p. 245]. Gödel made many statements expressing his preference for "computable" over "recursive" (see the quotes here from his collected works [Gödel, *1986*], [Gödel, *1990*], [Gödel, *1995*]), but neither Turing nor Gödel had much influence on the terminology of the subject after 1939.

The present terminology came from Church and Kleene. They had worked in the λ-definable functions until 1935 when they changed to recursive functions because it was more in the mathematical mainstream and had more audience appeal, as explained by Kleene *1987b*. They had both committed themselves to the new "recursive" terminology before they ever heard of Turing or his results. Furthermore, using "computable" in 1935 would not have increased audience appeal because a "computer" meant, even as late as 1946, a human being calculating with paper. Ironically, the personal computer revolution of the late 1970's which brought the technology, concept and terminology of computability to tens of millions arrived just as Kleene was retiring.

After 1938 Church had little influence on the subject or its terminology, although he did produce in the late 1940's and 1950's a number of students who later became quite prominent. Kleene, with his steady stream of papers giving fundamental tools like the hierarchies, normal form theorems, and recursion theorem (fixed point theorem) and opening new areas, dominated the subject from the late 1930's until at least the late 1950's, and his papers and book *1952* set the standard for the results and terminology, such as "recursive," "recursively enumerable," and "Church's Thesis." Post *1944* changed from his own terminology to that of Church and Kleene in his use of "recursive" and "recursively enumerable." The enormous popularity and influence of Post's paper and of Post's Problem firmly and widely established the Church-Kleene terminology. After the solution to Post's Problem by Friedberg and Muchnik in 1956-57 and the introduction of their priority method, the field greatly expanded, and there was no single dominant figure, but the existing terminology had been established and has continued to the present day.

4.6. Current usage of the concepts and terms. There is a current tendency in the subject to work in one formalism (usually that of Turing computable functions) but then to *name* the results using the terminology of *recursive* functions not *computable* functions. For example, consider from an *intensional viewpoint* the following quote from Putnam's recent review [1995, p. 371] of Roger Penrose's new book *1994*[12] about "a noncomputational ingredient in our conscious thinking."

> "First Penrose provides the reader with a proof of a form of the Gödel Theorem due to Alan Turing, the father of the modern digital computer and the creator of the mathematical subject *recursion theory*, which analyzes what computers can and cannot in principle accomplish."
> *—Hilary Putnam, review of Penrose 1994.*

[12]Physicist Penrose, like most scientists, never mentions the term "recursive," but he has an extensive discussion of Turing and Turing machines covering a whole chapter. Penrose [p. 66] writes, "by a computation (or algorithm) I indeed mean the action of some Turing machine, *i.e.*, in effect, just the operation of a computer according to some computer program." This is a good example of the acceptance in the scientific world of Turing's Thesis. (See §3).

There is very good reason to agree with Putnam[13] on his two assertions about Turing. However, Turing certainly *never* used the term "recursion theory" or "recursive function theory" for the subject. Turing mentioned the term "recursive function" only very briefly in *1937b* and [*1939*, §2] to say that these functions were mathematically equivalent to his Turing computable functions, and then Turing *dismissed* general recursive functions with the phrase, "we will not be much concerned here with this particular definition." Turing certainly never used "recursive" to mean "computable," and Turing did not refer to "recursive functions" again. Clearly from a *strictly intensional* viewpoint, the term *"recursion theory"* does not analyze anything about what *computers* can or cannot accomplish at all (contrary to Putnam's assertion); it deals with the properties of Herbrand-Gödel general recursive functions, the concepts of induction, recursion, reflexive program calls, and fixed points.

Gödel, who had invented *1934* the formal definition of general recursive function, abandoned it almost completely after seeing Turing machines *1936* and Turing's demonstration of Turing's Thesis. After 1936 Gödel rarely spoke of recursive functions, and never used the term "recursive" to mean "computable" or "decidable." Gödel often asserted later that Turing's was the correct definition of the notion of mechanical computability, and spoke often of the concept of computability *1946* [Gödel, *1946*, p. 84], *1951* [Gödel, *1995*, pp. 304–305], *193?* [Gödel, *1995*, p. 168], *1968* [Sieg, *1994*, p. 88], *1936* [Davis, *1965*, p. 82], *1964* [Davis, *1965*, p. 71–72].

Both Turing and Gödel, even later in life, rejected "recursive" as a name for the subject and often for their results. At his lecture *1949* on verifying program correctness, Turing used the term "induction variable," to which Prof. Hartree objected that the term should be "recursive variable" to distinguish it from the sense of mathematical induction. Turing [Turing, *1949*, p. 141] rejected the suggestion. In the three volumes of his collected works, [Gödel, *1986*], [Gödel, *1990*], and [Gödel, *1995*], Gödel *never* used the term "recursive function theory" to name the subject; when others did Gödel reacted sharply negatively, as related by Martin Davis.

> In a discussion with Gödel at the Institute for Advanced Study in Princeton about 1952–54, Martin Davis casually used the term "recursive function theory" as it was used then. Davis related, "To

[13]Putnam's *own article* [Putnam, *1995*] is an excellent example of the modern use of computer related concepts and terms rather than recursive functions to describe computational processes. Putnam's review (like Penrose's book) is written *entirely* using the *Turing machine model*, speaking of "machines," "programs," "output," "lines of code," a "debugged" program, "Turing-machine action," and "programs which output theorems." Putnam uses words like: "computer," "computational," "machine," and "program" over three dozen times, while "recursion" is mentioned only once (namely, in the quote above) and "primitive recursive" only once.

my surprise, Gödel reacted sharply, saying that the term in question should be used with reference to the kind of work Rosza Peter did."

In spite of the strong preference for "computable" by Turing and Gödel, the founders of the two formalisms and concepts, the name "recursive" instead of "computable" has been associated with almost all objects of the subject since the late 1930's. In spite of the computer revolution of the last few decades which Turing's work did so much to spawn, and which has given new connections between the subject and many outside areas in the scientific community, logicians have been slow to change the terminology and concepts of "recursive" to "computable." For various historical reasons there gradually emerged from 1936 to 1960 the following unspoken convention to use "recursive" as an all encompassing term for the concepts and for the name of the subject.

The *Recursion Convention* is to: (1) use the terms of the general recursive formalism (*i.e.*, "recursive," "recursively enumerable," "recursive in") to describe results about the subject, even if the proofs are based on the concepts and formalism of Turing computability; (2) use the term "*Church's Thesis*" to denote the amalgamation of the several theses, including theses by Church, Turing, and Post, in §2 and §3, even though Church's demonstration of his thesis (that all effectively calculable functions are general recursive) was flawed (§2.5) and was rejected by Gödel in its original form, and even though Turing gave an "unquestionably adequate" [Gödel's words] demonstration (§3.1) of Turing's Thesis (that all intuitively computable functions are Turing machine computable); (3) name the subject, and any new excursions such as to higher recursion, using the language of recursion, even if the concept of computability plays a very strong role there.

The Recursion Convention has been followed for over fifty years. Consider the following table of the basic *texts* on the subject, the *formalisms* that they use to define computable functions and relative computability, and the *names* that they assign afterward.

Here Turing computable and (general) recursive are as in Definition 1.2, register machines are in §3.5, and μ-recursive in §4.1. Of these texts no modern book (*i.e.*, after 1965) uses general recursive functions as the formalism for defining computable functions; two use μ-recursive functions (which is not the same as general recursive and was not used by Kleene to define "recursive") for ordinary computability, but then one (Lerman) changes to Turing machines for the more complicated case of relative computability, while the other (Odifreddi) stays with the μ-recursive definition of relative computability, but then gives a nonstandard proof of results about relative Turing computability, such as the Friedberg-Muchnik solution to Post's Problem. And yet all the authors (omitting Cutland who is writing a more elementary text for a general audience including computer scientists) use the *name* "recursive" for both the intuitive concept and the formally defined object, whether they have used a computability style definition or not.

Book	Defn of computable	Defn of relative computability	Name used for fn def'd
Kleene *1952*	gen'l recursive	gen'l recursive	recursive
Rogers *1967*	Turing machines	Turing machines	recursive
Cutland *1980*	register machines	register machines	computable
Lerman *1983*	μ-recursive	Turing machines	recursive
Soare *1987*	Turing machines	Turing machines	recursive
Odifreddi *1989*	μ-recursive	μ-recursive	recursive
Shoenfield *1991*	register machines	register machines	recursive

The Recursion Convention has brought "recursive" to have at least four different meanings as discussed in §5. This leads to some ambiguity. When a speaker uses the word "recursive" before a general audience, does he mean "defined by induction," "related to fixed points and reflexive program calls," or does he mean "computable?"

> The first rule of good taste in writing is to use words whose meaning will not be misunderstood; and if a reader does not know the meaning of the words, it is infinitely better that he should know he does not know it.
> —*Charles Sanders Peirce*, Ethics of Terminology, [Peirce, *1960*, p. 131].

Worse still, the Convention leads to *imprecise thinking* about the basic concepts of the subject; the term "recursion" is often used when the concept of "computability" is meant. (By the term "recursive function" does the writer mean "inductively defined function" or "computable function?") Furthermore, ambiguous and little recognized terms and imprecise thinking lead to *poor communication* both within the subject and to outsiders, which leads to isolation and lack of progress within the subject, since progress in science depends on the collaboration of many minds.

§5. Mathematical, scientific, and general English usage. The term "computable" appears as early as 1646 in English usage according to the *Oxford English Dictionary* (*O.E.D.*) *1989* [O.E.D., *1989*]. O.E.D. and *Webster's Third International Dictionary 1993* [Webster, *1993*] give the definition of "computable" as roughly synonymous with "calculable," capable of being ascertained or determined by a mathematical process especially of some intricacy. The meaning of "calculate" is somewhat more general including "to figure out," "to design or adapt for a purpose," "to judge to be probable," while "compute" means more "to determine by a mathematical process," or "to determine or calculate by means of a computer."

When Dedekind *1888* proved that a definition by recursion uniquely defines a function, he called it "definition by induction." Hilbert *1904* used the term "*rekurrent(e),*" and in *1923* he used "*Rekursion*". The term "*recursive*" was apparently first used in English by Ramsey *1928* (see Gandy [Gandy, *1988*, p. 73]). Skolem in *1923* showed that many number-theoretic functions are primitive recursive, and he used "*rekurrierend.*" In *1926* Hilbert expanded the use of the term to include transfinite types and essentially transfinite recursion. Ackermann *1928* considered functions which can be defined using primitive recursion at all finite types. He gave a definition of a particular function using double nested recursion and showed that it was not primitive recursive. R. Péter *1934* and *1951* examined primitive recursive functions and special recursive functions (where recursion on more than one variable is allowed).

The current meanings of "recursive" derive from the verb *"recur"* which means to return to a place or status, or the concept of "*definition by recursion*", like Scheme (V) in (1), for which Webster gives the meaning: a definition of a function permitting values of the function to be calculated systematically in a finite number of steps, especially a mathematical definition in which the first case is given and the nth case is defined in terms of one or more previous cases, especially the immediately preceding one. Thus, the term "*recursive*" is presently used in the subject in at least *four* different ways which we now summarize as a definition for future reference.

DEFINITION 5.1. The current meanings of *recursive* and *recursion* are these:
(i) *recursion* is used with meanings derived from the verb "recur," as in the dictionary definition of "recursion" above;
(ii) *recursion* is used in the sense of "definition by recursion" (*i.e.*, definition by induction) as defined in equation (1) of §2.1 and in the dictionary entry of definition by recursion above;
(iii) following Kleene *1936* and Church *1936* the term "*recursive*" denotes "general recursive" and any of its mathematically equivalent *formal* variants, such as "Turing computable," "λ-definable," "specified by a Post *1944* normal system," or Kleene's "μ-recursive".
(iv) "*recursive*" is used to mean any of the *informal* variants of Definition 1.1 such as "(intuitively) computable," "effectively calculable," "defined by a mechanical process," or "specified by an algorithm."

Most dictionaries give meaning (i) and usually (ii). Most people outside the subject including computer scientists, mathematicians, and scientists understand "recursive" as (ii) if they know it at all. Of the dictionaries only O.E.D. gives meaning (iii) and then only in the fine print, without a definition, but with reference to Kleene *1936* and *1952*, an entry written by Gandy. *None* of these dictionaries gives meaning (iv) that "recursive" means "computable" or "decidable." This is a meaning understood by very few outside the subject.

§6. Themes and goals of computability theory. Many believe that the present subject of recursion theory would benefit from: (1) the pruning of some more technical and specialized topics while retaining most of the present research content; (2) a broadening of horizons and problems to others areas in logic, mathematics, computer science, and science in general in interaction with computability; (3) a better communication of present and future results in both (1) and (2) in terms of some of the basic concepts below to the larger scientific community.[14] Before presenting his lecture or paper, the author should ask himself, "What light does this shed on the basic themes and goals of the subject such as computability, enumerability, information content, relation to other branches of logic and mathematics?" As Harvey Friedman has suggested, every morning one should wake up and reflect on the conceptual and foundational significance of one's work. This re-evaluation process should be carried out *regardless of names* for the subject.

The following items should be considered a mixture of concepts, goals, themes, and connections with other areas: computability; enumerability; relative computability (Turing reducibility, $A \leq_{\mathrm{T}} B$); information content, normally measured by Turing degree; computational complexity and computing with bounds on resources of space and time; polynomial hierarchy questions; definability; invariance and automorphisms; elementary theory; relationship of computability, enumerability, and information content to algebraic structures; relationship of computability to model theory, and set theory, and proof theory, for example: models of arithmetic; provably computable functions, reverse mathematics and levels in the arithmetic hierarchy; relationship of computability to topology, to algebra and combinatorics, to analysis, *e.g.*, to descriptive set theory; relationship to number theory (*e.g.*, Hilbert's 10th Problem); relationship of computability to computer science; Kleene arithmetic hierarchy and the Meyer-Stockmeyer hierarchy for polynomial reducibility, structures in complexity; relationship of computability to other fields, *e.g.*, biology, quantum physics, economics, etc.

§7. Analysis. Both of the concepts of recursion and computability have played a crucial role in the development of the subject and will continue to do so.

[14]Of course, a desire by its proponents to improve a field is not an indication that it is in greater need of improvement than other fields, but rather indicates the intention to strengthen it still further.

The term and concept of "computable" is associated with the notion of computation (§2.1), algorithm (§2.3), and with the functions defined by (or sets enumerated by) Turing machines (§3.1) or register machines (§3.4), and also with relative Turing computability (§4.3).

The term and concept of "recursive" is associated with: definition by recursion (induction) (§2.2), general recursive functions in the sense in Herbrand-Gödel (§2.4), fixed points as in the Kleene Recursion Theorem or more generally Kleene's schema (S11) (see *1987c*), which Kleene believed included all possible recursions, and Kleene's μ-recursive functions (§4.1).

RECOMMENDATION 1. *The term "recursive" should no longer carry the additional meaning of "computable" or "decidable."*

We already have better words to convey these meanings, namely "computable" and "decidable" themselves. Likewise, the term "computable" should not carry the extra meaning "recursive." Recommendation 1 will have several advantages. First, we will be able to recognize clearly the concept of computability when it appears. Second, it reinforces the original meaning of "recursive" and induction as understood by Dedekind *1888*, Peano *1889* and *1891*, Hilbert *1904* and *1926*, Skolem *1923*, Gödel *1931* and *1934*, and Péter *1934* and *1951*, and most modern computer scientists, mathematicians, and physical scientists, and as expanded to fixed points, the recursion theorem, and to other kinds of recursion by Kleene, Platek, and others.

RECOMMENDATION 2. *If functions are defined, or sets are enumerated, or relative computability is defined using Turing machines, register machines, or variants of these (as in the texts in the table of* §4.6 *or in the Putnam* 1995 *review), then the name "computable" rather than "recursive" should be attached to the result, as in Cutland* 1980*, Davis* 1958*, as well as* [Boolos and Jeffrey, *1974*] *the subtitle of* [Soare, *1987*] *and others.*

Thus, the terms "recursive" and "computable" would reacquire their traditional and original meanings, and those understood by most outsiders (§5). This is in accord with the usage and opinion of the founders of the two concepts and terms, Turing and Gödel, both of whom used "computability" in this sense and both of whom rejected the use of "recursive" to mean computable (§4.6).

This would improve communication with many researchers outside the field. It would also give a new scientific precision to our discussions within the subject and would remove various ambiguities mentioned above. (For example, by a "recursive function" do we mean computable one or one defined by induction?) It would enable us to speak with greater clarity and precision about our own subject from r.e. sets (á la Post §4.2) to computability in higher types where the relative role of the two concepts has always been controversial, even to the experts (see §4.4).

RECOMMENDATION 3. *We should distinguish between the intensional meaning of Church's Thesis* (*that all effectively calculable functions are general recursive*) *versus Turing's Thesis* (*that all intuitively computable functions are computable by a Turing machine*). *When we are writing a paper dealing with which classes of functions are Turing computable* (i.e., *mechanistic*), *as in Gandy* 1980 *and in many other places, we should refer to "*Turing's Thesis*"* (*as in Sieg* 1994 *and Tamburrini* 1995) *not to "Church's Thesis."*

RECOMMENDATION 4. *We should call the subject "*Computability Theory*" or simply* Computability *instead of* Recursive Function Theory *or* Recursion Theory.

In 1935 when Church and Kleene adopted the terminology and formalism of "recursive functions" for the subject in preference to λ-definable functions, there was good reason to do so. Recursive functions lay in an established mathematical tradition and had more appeal to a general mathematical audience. Church was eager to establish mathematical logic as a field of mathematics and to avoid the inconsistencies discovered by Kleene and Rosser in Church's general λ-system. When Kleene and Church saw Turing's model in 1936, they had little desire to make another change so soon. Another conversion to the formalism of Turing would have had little increase in appeal to a general audience because as late as 1946 the term "computer" meant a human being calculating with pencil and paper. It was only after 1980 (about when Kleene retired) that the revolution in personal computers made the terminology and concepts of computing household items, so that now they are much better known to the public than those of recursion. Some mathematicians now argue that we have been using "recursive" for sixty years to mean "computable" and "effective," and we should continue for the sake of tradition; besides names do not matter in mathematics anyway.

In contrast, many mathematicians and philosophers of science, including Alfred Tarski, Charles Sanders Peirce, and Georg Cantor believed strongly that terms and names are very important to the advancement of a field.

> "I am extremely careful with the choice of those [i.e. new notions], as I take the position that the development and propagation in no small degree depends on a fortunate and properly fitting terminology."
> —*Georg Cantor*

Philosopher Charles S. Peirce described the importance of language for science this way.

> "the woof and warp of all thought and all research is symbols, and the life of thought and science is the life inherent in symbols; so that

> it is wrong to say that a good language is *important* to good thought, merely; for it is of the essence of it."
> —*Charles Sanders Peirce*, The Ethics of Terminology, Volume II *Elements of Logic* in: [Peirce, *1960*, p. 129].

The supporters of "*Computability Theory*" believe that: (1) the concept (as developed in §2.1 and §3.1) of computability is the *heart of the subject* in virtually all its branches; other concepts (recursion, definability) are very important but not at the center; (2) "computability" is *historically more accurate* because Gödel and Turing, the founders of the two formalisms, both overwhelmingly preferred the concept and name "computability" to describe the work and the subject (§4.6). Hence, the current proposed name is not a *change* but a *return* to the original terminology of the founders of the terms, Gödel and Turing, a terminology which somehow became lost.

Using "computable" instead of "recursive" in names and terms would (3) *improve communication of results* within the field and to nonexperts and of questions flowing in. The term "computable" is listed in virtually all dictionaries as a synonym for "calculable." When "recursive" appears in dictionaries (§5) it is with meanings related to "recur," or "induction," very rarely with the meaning of "general recursive" and never with the meaning of "computable" or "decidable." Therefore, when we use "recursive" to mean "computable," we are using it in a way that is not in any dictionary and which an educated scholar or scientist cannot reasonably be expected to know. Indeed, there is a danger that a computer scientist or mathematician might mistake it for "inductive."

(Of course, dictionaries do not determine the meaning of mathematical terms; they merely reflect the public understanding of those terms which mathematicians choose to use. However, now sixty years after the first use of "recursive" to mean "computable," the dictionaries and general public today still have not caught on. In contrast, mathematical terms like "homotopy" are listed in O.E.D. with the usual meaning from topology.)

The importance of improved communication between those within and outside the subject due to improved, more accurate, and commonly understood terminology should not be underestimated.

> Thirdly, the progress of science cannot go far except by collaboration; or, to speak more accurately, no mind can take one step without the aid of others minds.
> —*Charles Sanders Peirce*, Ethics of Terminology, [Peirce, *1960*, p. 129].

The name "*Computability Theory*" could be construed (4) as an *affirmation of Turing's Thesis* since it suggests to the outside world that the (Turing) computable functions which we *actually* study coincide with the intuitively computable ones as the nonspecialists understand the term "computable." Since virtually all practitioners of Computability Theory strongly accept

Turing's Thesis, this is a desirable consequence. In contrast, the present name "Recursion Theory" suggests to the nonexpert that the subject is about functions defined by induction and suggests nothing about computability, which is misleading of the real content of the subject.

Next, (5) there are the *precedents of previous name changes* for parts or all of the subject. In the 1970's, ω-computability theory (computability on ω) was called "*ordinary recursion theory* (*O.R.T.*)" to distinguish it from "*generalized recursion theory* (*G.R.T.*)." In lectures at C.I.M.E. Bressanone, Italy, *1979*, again at the A.M.S. Institute at Cornell in 1982, and in his book Soare [*1987*, pp. x–xi] suggested changing the name to "*classical recursion theory* (*C.R.T.*)," and justified the term by analogy with the qualities of classical art of the Renaissance (see *1981*). Odifreddi *1989* adopted the phrase as the title for his book, and it is now the accepted name for this branch of the subject, *e.g.*, in Sacks *1990*. In the early 1930's Church, Kleene, and Rosser had carried out all their research in the formalism of Church's λ-calculus. In 1935 and 1936 Church and Kleene changed their formalism and terminology from λ-definable functions to (general) recursive ones because the latter had more appeal to the audiences and was more in an established tradition and terminology which the audiences recognized (§4.5). Reasons for the *present* change include these two and many more.

Many believe that the computability terminology, and especially the *concepts* it will stress, will (6) *improve the content* of the subject and *facilitate the goals* of §6 as did the Church-Kleene change away from λ-definable functions in 1935. That change introduced an entirely *new* formalism into the subject, while here the name change will more accurately describe the formalism we are *already* using (see the table in §4.6) under the name of recursion.

§8. Conclusion. Gödel noticed that there is an absoluteness to computability rarely found elsewhere. Specifically, Gödel was referring to a sequence of formal systems $S_i \subseteq S_{i+1}$, $1 \leq i$, where S_1 includes, say Peano arithmetic. Gödel noticed that a statement like $\varphi(n) = m$ might have a much shorter proof in S_{i+1} than in S_i by adding additional axioms. However Gödel remarked,

> "It may be shown that a function which is computable in one of the systems S_i or even in a system of transfinite type, is already computable in S_1. Thus, the concept "computable" is in a certain definite sense "absolute," while practically all other familiar metamathematical concepts depend quite essentially on the system with respect to which they are defined."
>
> —*Gödel: On the length of proofs, 1936,* [Davis, *1965*, p. 82].

Since this subject deals with relative computability and since the least class under $\leq_T$ (the class of computable sets) is trivial, it has often been remarked that the subject is about noncomputable objects more than computable ones. (Indeed most of mathematics is about noncomputable objects and even uncountable ones.) A major objective in the subject is to classify noncomputable objects under relative computability. To do this one mainly constructs a Turing reduction Φ to show that one set B is computable in another set A, namely $\Phi(A) = B$, or else proves that no such Φ can exist. However, it is well known, see Soare [*1987*, p. 86], that building any such Φ is equivalent to constructing a computably enumerable set, V_Φ, of axioms $\langle \sigma, x, y \rangle$, $\sigma \in 2^{<\omega}$, such that $\Phi^\sigma(x) = y$. Hence, the Gödel absoluteness is preserved in relative computability as well as computability. The case of B being computable, namely $B \leq_{\mathrm{T}} \emptyset$, is just a special case.

The changes recommended in this paper are intended to restore and enhance the beauty of the subject by returning to the clarity of the conception of computability envisioned by Turing and Gödel.

REFERENCES

[Boolos and Jeffrey, *1974*] G. Boolos and R. Jeffrey, ***Computability and logic***, Cambridge University Press, Cambridge, England.

[Church, *1935*] A. Church, *An unsolvable problem of elementary number theory*, ***Bulletin of the American Mathematical Society***, vol. 41, pp. 332–333, Preliminary report (abstract).

[Church, *1936*] ———, *An unsolvable problem of elementary number theory*, ***American Journal of Mathematics***, vol. 58, pp. 345–363.

[Church, *1936b*] ———, *A note on the Entscheidungsproblem*, ***Journal of Symbolic Logic***, vol. 1, pp. 40–41, correction 101–102.

[Church, *1937*] ———, *Review of Turing* 1936, ***Journal of Symbolic Logic***, vol. 2, no. 1, pp. 42–43.

[Church, *1937b*] ———, *Review of Post* 1936, ***Journal of Symbolic Logic***, vol. 2, no. 1, p. 43.

[Church, *1938*] ———, *The constructive second number class*, ***Bulletin of the American Mathematical Society***, vol. 44, pp. 224–232.

[Church and Kleene, *1936*] A. Church and S. C. Kleene, *Formal definitions in the theory of ordinal numbers*, ***Fundamenta Mathematicae***, vol. 28, pp. 11–21.

[Cutland, *1980*] Nigel Cutland, ***Computability: An introduction to recursive function theory***, Cambridge University Press, Cambridge, England.

[Davis, *1958*] M. Davis, ***Computability and unsolvability***, Mc-Graw-Hill, New York, reprinted in 1982 by Dover Publications.

[Davis, *1965*] M. Davis (editor), ***The undecidable. Basic papers on undecidable propositions, unsolvable problems, and computable functions***, Raven Press, Hewlett, New York.

[Davis, *1982*] ———, *Why Gödel did not have Church's thesis*, ***Information and Control***, vol. 54, pp. 3–24.

[Davis, *1988*] ———, *Mathematical logic and the origin of modern computers*, in Herken, *1988*, pp. 149–174.

[Dedekind, *1872*] R. Dedekind, ***Stetigkeit und irrational Zahlen***, Braunschweig, 5th ed., 1927; also in ***Dedekind Gesammelte mathematische Werke*** (Vieweg and Sohn, editors), vol. III, Braunschweig, 1932, pp. 315–334; English translation by Wooster Woodruff Beman entitled *Continuity and irrational numbers*, ***Essays on the theory of numbers***, Open Court, Chicago, 1901, pp. 1–24.

[Dedekind, *1888*] ———, ***Was sind und was sollen die Zahlen?***, Braunschweig, 6th ed., 1930; also in ***Dedekind Gesammelte Mathematische Werke***, vol. III, pp. 335–391; English translation by Beman, ***The nature and meaning of numbers. loc. cit.***, pp. 31–105; English translation in Dedekind, ***Essays on the theory of numbers***, Open Court, Chicago, 1901, pp. 29–115.

[Epstein and Carnielli, *1989*] R. L. EPSTEIN and W. A. CARNIELLI, ***Computability: computable functions, logic, and the foundations of mathematics***, Brooks/Cole Advanced Books and Software, Pacific Grove, Calif.

[Feferman, *1988*] S. FEFERMAN, *Turing in the land of* $O(z)$, in Herken, *1988*, pp. 113–147.

[Feferman, *1992*] ———, *Turing's "oracle": From absolute to relative computability—and back*, ***The space of mathematics*** (J. Echeverria et al., editors), Walter de Gruyter, Berlin, pp. 314–348.

[Fitting, *1987*] M. FITTING, ***Computability theory, semantics, and logic programming***, Oxford University Press.

[Gandy, *1980*] R. GANDY, *Church's thesis and principles for mechanisms*, ***The Kleene symposium***, North-Holland, pp. 123–148.

[Gandy, *1988*] ———, *The confluence of ideas in 1936*, in Herken, *1988*, pp. 55–111.

[Gödel, *193?*] K. GÖDEL, *Undecidable diophantine propositions*, in Gödel *1995*, pp. 156–175.

[Gödel, *1931*] ———, *Über formal unentscheidbare sätze der Principia Mathematica und verwandter systeme. I*, ***Monatsch. Math. Phys.***, vol. 38, pp. 173–178, English translation in Davis *1965*, pp. 4–38, and in van Heijenoort, *1967*, pp. 592–616.

[Gödel, *1934*] ———, *On undecidable propositions of formal mathematical systems*, notes by S. C. Kleene and J. B. Rosser on lectures at the Institute for Advanced Study, Princeton, New Jersey, 1934, reprinted in Davis *1965*, pp. 39–74].

[Gödel, *1936*] ———, *On the length of proofs*, Gödel *1986*, pp. 397–399; reprinted in Davis *1965*, pp. 82–83, with a *Remark* added in proof [of the original German publication].

[Gödel, *1946*] ———, *Remarks before the Princeton bicentennial conference of problems in mathematics*, reprinted in Davis *1965*, pp. 84–88.

[Gödel, *1951*] ———, *Some basic theorems on the foundations of mathematics and their implications*, in Gödel *1995*, pp. 304–323. (This was the Gibbs Lecture delivered by Gödel on December 26, 1951 to the American Mathematical Society).

[Gödel, *1958*] ———, *Über eine bisher noch nicht benütze Erweiterung des finiten Standpunktes*, ***Dialectica***, vol. 12, pp. 280–287, German and English translation in Godel *1986*, pp. 240–251, with introductory note by A. S. Troelstra, pp. 217–241.

[Gödel, *1964*] ———, *Postscriptum to Gödel* 1931, written in 1946, printed in Davis *1965*, pp. 71–73.

[Gödel, *1972*] ———, *Some remarks on the undecidability results*, in Gödel *1990*, pp. 305–306, written in 1972.

[Gödel, *1986*] ———, ***Collected works volume I: Publications 1929–36*** (S. Feferman et al., editors), Oxford University Press, Oxford.

[Gödel, *1990*] ———, ***Collected works volume II: Publications 1938–1974*** (S. Feferman et al., editors), Oxford University Press, Oxford.

[Gödel, *1995*] ———, ***Collected works volume III: Unpublished essays and lectures*** (S. Feferman et al., editors), Oxford University Press, Oxford.

[Harrington and Soare, *1996*] L. HARRINGTON and R. I. SOARE, *Definability, automorphisms, and dynamic properties of computably enumerable sets*, this BULLETIN, vol. 2, 1996, pp. 199–213.

[Herken, *1988*] R. Herken (editor), ***The universal Turing machine: A half-century survey***, Oxford University Press.

[Hilbert, *1899*] D. HILBERT, ***Grundlagen der Geometrie***, 7th ed., Tuebner-Verlag, Leipzig, Berlin, 1930.

[Hilbert, *1904*] ———, *Über die Grundlagen der Logik und der Arithmetik*, in ***Verhandlungen des Dritten Internationalen Mathematiker-Kongresses in Heidelberg vom 8. bis 13. August 1904***, Teubner, Leipzig, 1905, pp. 174–185. Reprinted in van Heijenoort *1967*, pp. 129–138.

[Hilbert, *1918*] ———, *Axiomatisches Denken*, ***Mathematische Annalen***, vol. 78, pp. 405–415.

[Hilbert, *1926*] ———, *Über das Unendliche*, ***Mathematische Annalen***, vol. 95, pp. 161–190, English translation in van Heijenoort *1967*, 367–392.

[Hilbert, *1928*] ———, *Abhandlungen aus dem mathematischen Seminar derHamburgischen Universität*, ***Die Grundlagen der Mathematik***, vol. 6, pp. 65–85, reprinted in van Heijenoort *1967*, pp. 464–479.

[Hilbert and Ackermann, *1928*] D. HILBERT and W. ACKERMANN, ***Grundzüge der theoretischen Logik***, Springer-Verlag, Berlin, English translation of 1938 edition, Chelsea, New York, 1950.

[Hilbert and Bernays, *1934*] D. HILBERT and P. BERNAYS, ***Grundlagen der Mathematik I (1934), II (1939)***, second ed., Springer-Verlag, Berlin, I (1968), II (1970).

[Hodges, *1983*] A. HODGES, ***Alan Turing: The enigma***, Burnett Books and Hutchinson, London, and Simon and Schuster, New York.

[Kleene, *1936*] S. C. KLEENE, *General recursive functions of natural numbers*, ***Mathematische Annalen***, vol. 112, pp. 727–742.

[Kleene, *1936b*] ———, *λ-definability and recursiveness*, ***Duke Mathematics Journal***, vol. 2, pp. 340–353.

[Kleene, *1936c*] ———, *A note on recursive functions*, ***Bulletin of the American Mathematical Society***, vol. 42, pp. 544–546.

[Kleene, *1938*] ———, *On notation for ordinal numbers*, ***Journal of Symbolic Logic***, vol. 3, pp. 150–155.

[Kleene, *1943*] ———, *Recursive predicates and quantifiers*, ***Transactions of the American Mathematical Society***, vol. 53, pp. 41–73.

[Kleene, *1944*] ———, *On the forms of the predicates in the theory of constructive ordinals*, ***American Journal of Mathematics***, vol. 66, pp. 41–58.

[Kleene, *1952*] ———, ***Introduction to metamathematics***, Van Nostrand, New York, ninth reprint 1988, Walters-Noordhoff Publishing Co., Groningën and North-Holland, Amsterdam.

[Kleene, *1952b*] ———, *Recursive functions and intuitionistic mathematics*, ***Proceedings of the international congress of mathematicians, Cambridge, Massachusetts, U.S.A., August 30–September 6, 1950***, vol. 1, American Mathematical Society (Providence, R.I.), pp. 679–685.

[Kleene, *1955*] ———, *Arithmetical predicates and function quantifiers*, ***Transactions of the American Mathematical Society***, vol. 79, pp. 312–340.

[Kleene, *1955b*] ———, *On the forms of the predicates in the theory of constructive ordinals* (*second paper*), ***American Journal of Mathematics***, vol. 77, pp. 405–428.

[Kleene, *1955c*] ———, *Hierarchies of number-theoretical predicates*, ***Bulletin of the American Mathematical Society***, vol. 61, pp. 193–213.

[Kleene, *1959*] ———, *Recursive functionals and quantifiers of finite type I*, ***Transactions of the American Mathematical Society***, vol. 91, pp. 1–52.

[Kleene, *1962*] ———, *Turing-machine computable functionals of finite types I*, ***Logic, methodology, and philosophy of science: Proceedings of the 1960 international congress***, Stanford University Press, pp. 38–45.

[Kleene, *1962b*] ———, *Turing-machine computable functionals of finite types II*, ***Proceedings of the London Mathematical Society***, vol. 12, no. 3, pp. 245–258.

[Kleene, *1963*] ———, *Recursive functionals and quantifiers of finite type II*, ***Transactions of the American Mathematical Society***, vol. 108, pp. 106–142.

[Kleene, *1981*] ———, *Origins of recursive function theory*, ***Annals of the History of Computing***, vol. 3, pp. 52–67.

[Kleene, *1981b*] ———, *The theory of recursive functions, approaching its centennial*, ***Bulletin of the American Mathematical Society (n.s.)***, vol. 5, pp. 43–61.

[Kleene, *1981c*] ———, *Algorithms in various contexts*, ***Proc. sympos. algorithms in modern mathematics and computer science (dedicated to Al-Khowarizimi), Urgench, Khorezm Region, Uzbek, SSSR, 1979***, Springer-Verlag, Berlin, Heidelberg and New York.

[Kleene, *1987*] ———, *Reflections on Church's Thesis*, ***Notre Dame Journal of Formal Logic***, vol. 28, pp. 490–498.

[Kleene, *1987b*] ———, *Gödel's impression on students of logic in the 1930's*, ***Gödel remembered*** (P. Weingartner and L. Schmetterer, editors), Bibliopolis, Naples, pp. 49–64.

[Kleene, *1988*] ———, *Turing's analysis of computability, and major applications of it*, in Herken *1988*, pp. 17–54.

[Kleene-Post, *1954*] S. C. KLEENE and E. L. POST, *The upper semi-lattice of degrees of recursive unsolvability*, ***Annals of Mathematics***, vol. 59, pp. 379–407.

[Kline, *1972*] M. KLINE, ***Mathematical thought from ancient to modern times***, Oxford University Press, Oxford.

[Lerman, *1983*] M. LERMAN, ***Degrees of unsolvability: Local and global theory***, Springer-Verlag, Heidelberg, New York, Tokyo.

[Normann, *1980*] D. NORMANN, *Recursion on countable functionals*, ***Lecture notes in mathematics***, no. 811, Springer-Verlag, Heidelberg, New York.

[Odifreddi, *1989*] P. ODIFREDDI, ***Classical recursion theory***, North-Holland, Amsterdam.

[O.E.D., *1989*] J. A. Simpson and E. S. C. Weiner (editors), ***Oxford english dictionary***, second ed., Clarendon Press, Oxford.

[Peano, *1889*] G. PEANO, ***Arithmetices principia, nova methodo exposita***, Turin Bocca, English translation in van Heijenoort, *1967*, pp. 83–97.

[Peano, *1891*] ———, *Sul concetto di numero*, ***Rivista di Matematica***, vol. 1, pp. 87–102, 256–267.

[Peirce, *1960*] CHARLES S. PEIRCE, *Book II. Speculative grammar*, ***Volume II: Elements of logic*** (C. Hartshorne and P. Weiss, editors), ***Collected Papers of Charles Sanders Peirce***, The Belknap Press of Harvard University Press, Cambridge, Massachusetts and London, England.

[Penrose, *1994*] R. PENROSE, ***Shadows of the mind***, Oxford University Press, Oxford.

[Peter, *1934*] R. PÉTER, *Über den Zussammenhang der verschiedenen Begriffe der rekursiven Funktion*, ***Mathematische Annalen***, vol. 110, pp. 612–632.

[Peter, *1951*] ———, ***Rekursive funktionen***, Akadémaiai Kiadó (Akademische Verlag), Budapest, ***Recursive functions***, third revised ed., Academic Press, New York, 1967.

[Platek, *1966*] R. PLATEK, *Foundations of recursion theory*, ***Ph.D. thesis***, Stanford University, Stanford, CA.

[Post, *1936*] E. L. POST, *Finite combinatory processes–formulation I*, ***Journal of Symbolic Logic***, vol. 1, pp. 103–105, reprinted in Davis *1965*, pp. 288–291.

[Post, *1941*] ———, *Absolutely unsolvable problems and relatively undecidable propositions: Account of an anticipation*, submitted for publication in 1941; printed in Davis *1965*, pp. 340–433.

[Post, *1943*] ———, *Formal reductions of the general combinatorial decision problem*, ***American Journal of Mathematics***, vol. 65, pp. 197–215.

[Post, *1944*] ———, *Recursively enumerable sets of positive integers and their decision problems*, ***Bulletin of the American Mathematical Society***, vol. 50, pp. 284–316, reprinted in Davis *1965*, pp. 304–337.

[Post, *1947*] ———, *Recursive unsolvability of a problem of Thue*, ***Journal of Symbolic Logic***, vol. 12, pp. 1–11, reprinted in Davis *1965*, pp. 292–303.

[Post, *1948*] ———, *Degrees of recursive unsolvability: preliminary report (abstract)*, ***Bulletin of the American Mathematical Society***, vol. 54, pp. 641–642.

[Putnam, *1995*] H. PUTNAM, *Review of Penrose* 1994, ***Bulletin of the American Mathematical Society***, vol. 32, pp. 370–373.

[Sacks, *1990*] G. E. SACKS, ***Higher recursion theory***, Springer-Verlag, Heidelberg, New York.

[Shoenfield, *1967*] J. R. SHOENFIELD, ***Mathematical logic***, Addison-Wesley, Reading, Massachusetts.

[Shoenfield, *1991*] ———, *Recursion theory*, ***Lecture notes in logic***, Springer-Verlag, Heidelberg, New York.

[Shoenfield, *1995*] ———, *The mathematical work of S. C. Kleene*, this BULLETIN, vol. 1, pp. 8–43.

[Sieg, *1994*] W. SIEG, *Mechanical procedures and mathematical experience*, ***Mathematics and mind*** (A. George, editor), Oxford University Press.

[Sieg and Byrnes, *1995*] W. SIEG and J. BYRNES, *K-graph machines: generalizing Turing's machines and arguments*, preprint.

[Skolem, *1923*] T. SKOLEM, *Begründung der elementaren Arithmetik durch die rekurrierende Denkweise ohne Anwendung scheinbare Veränderlichen mit unendlichen Ausdehnungsbereich*, ***Skrifter utgit av Videnskapsselskapet i Kristiania, I. Mathematisk-Naturvidenskabelig Klasse***, vol. 6, English translation in van Heijenoort, *1967*, pp. 302–333.

[Soare, ta1] R. I. SOARE, *An overview of the computably enumerable sets*, ***Handbook of computability theory***, North-Holland, Amsterdam, in preparation.

[Soare, ta2] ———, *Computability and enumerability*, ***Proceedings of the 10th international congress of logic, methodology, and philosophy of science, August 19–25, 1995, Florence, Italy***, to appear.

[Soare, *1981*] ———, *Constructions in the recursively enumerable degrees*, ***Recursion theory and computational complexity*** (G. Lolli, editor), Proceedings of Centro Internazionale Matematico Estivo (C.I.M.E.), June 14–23, 1979, in Bressanone, Italy, Liguori Editore, Naples, Italy.

[Soare, *1987*] ———, ***Recursively enumerable sets and degrees: A study of computable functions and computably generated sets***, Springer-Verlag, Heidelberg.

[Tamburrini, *1995*] G. TAMBURRINI, *Mechanistic theories in cognitive science: The import of Turing's Thesis*, ***Proceedings of the 10th international congress of logic, methodology, and philosophy of science, August 19–25, 1995, Florence, Italy***, to appear.

[Turing, *1936*] A. M. TURING, *On computable numbers, with an application to the Entscheidungsproblem, parts 3 and 4*, ***Proceedings of the London Mathematical Society***, vol. 42, pp. 230–265; [Turing, *1937*] *A correction*, ibid., vol. 43, pp. 544–546[15] .

[15] Many papers, Kleene [*1943*, p. 73], *1987*, *1987b*, Davis [*1965*, p. 72], Post [*1943*, p. 200], and others, mistakenly refer to this paper as "[Turing, *1937*]," perhaps because the volume **42** is 1936-37 covering 1936 and part of 1937, or perhaps because of the two page minor correction *1937*. Others, such as Kleene *1952*, *1981*, *1981b*, Kleene and Post [*1954*, p. 407], Gandy *1980*, Cutland *1980*, and others, correctly refer to it as "[*1936*]," or sometimes "[*1936-37*]." The journal states that Turing's manuscript was "Received 28 May, 1936–Read 12 November, 1936." It appeared in two sections, the first section of pages 230–240 in Volume **42**, Part 3, issued on November 30, 1936, and the second section of pages 241–265 in Volume

[Turing, *1937b*] ———, *Computability and λ-definability*, ***Journal of Symbolic Logic***, vol. 2, pp. 153–163.

[Turing, *1939*] ———, *Systems of logic based on ordinals, part 3*, ***Proceedings of the London Mathematical Society***, vol. 45, pp. 161–228, reprinted in Davis *1965*, pp. 154–222.

[Turing, *1948*] ———, *Intelligent machinery*, ***Machine Intelligence***, vol. 5, pp. 3–23, written in September, 1947 and submitted to the National Physical Laboratory in 1948.

[Turing, *1949*] ———, *Text of a lecture by Turing on June 24, 1949*, in F. L. Morris and C. B. Jones, *An early program proof by Alan Turing*, ***Annals of the History of Computing***, vol. 6 (1984), pp. 139–143.

[Turing, *1950*] ———, *Computing machinery and intelligence*, ***Mind***, vol. 59, pp. 433–460.

[Turing, *1950b*] ———, *The word problem in semi-groups with cancellation*, ***Annals of Mathematics***, vol. 52, pp. 491–505.

[Turing, *1954*] ———, *Solvable and unsolvable problems*, ***Science News***, vol. 31, pp. 7–23.

[Turing, *1986*] ———, *Lecture to the London Mathematical Society on 20 February 1947*, ***A. M. Turing's ACE report of 1946 and other papers*** (B. E. Carpenter and R. W. Doran, editors), Cambridge University Press, pp. 106–124.

[van Heijenoort, *1967*] J. van Heijenoort (editor), ***From Frege to Gödel, a sourcebook in mathematical logic, 1879–1931***, Harvard University Press, Cambridge, Massachusetts.

[Webster, *1993*] Ph. B. Gove (editor), ***Webster's third new international dictionary of the English language (unabridged)***, Merriam-Webster Inc., Publishers, Springfield, Massachusetts, U.S.A.

[Zabell, *1995*] S. L. Zabell, *Alan Turing and the central limit theorem*, ***American Mathematical Monthly***, vol. 102, no. 6, pp. 483–494.

DEPARTMENT OF MATHEMATICS
UNIVERSITY OF CHICAGO
CHICAGO, ILLINOIS 60637-1546, USA

E-mail: soare@math.uchicago.edu
http://www.cs.uchicago.edu/~soare
Papers posted at ftp:cs.uchicago.edu/ftp/pub/users/soare

42, Part 4, issued December 23, 1936. No part of Turing's paper appeared in 1937, but the two page minor correction *1937* did. Determining the correct date of publication of Turing's work is important to place it chronologically in comparison with Church *1936*, Post *1936*, and Kleene *1936*.