

8-1-2005

Church Without Dogma: Axioms for Computability

Wilfried Sieg
Carnegie Mellon University

Follow this and additional works at: <http://repository.cmu.edu/philosophy>



Part of the [Philosophy Commons](#)

Recommended Citation

Sieg, Wilfried, "Church Without Dogma: Axioms for Computability" (2005). *Department of Philosophy*. Paper 119.
<http://repository.cmu.edu/philosophy/119>

This Technical Report is brought to you for free and open access by the Dietrich College of Humanities and Social Sciences at Research Showcase @ CMU. It has been accepted for inclusion in Department of Philosophy by an authorized administrator of Research Showcase @ CMU. For more information, please contact research-showcase@andrew.cmu.edu.

CHURCH WITHOUT DOGMA:

Axioms for computability

Wilfried Sieg

August 1, 2005

Technical Report No. CMU-PHIL-175

Philosophy

Methodology

Logic

Carnegie Mellon

Pittsburgh, Pennsylvania 15213

CHURCH WITHOUT DOGMA:

Axioms for computability

Wilfried Sieg
Carnegie Mellon University
Pittsburgh

Abstract: Church's and Turing's theses dogmatically assert that an informal notion of effective calculability is adequately captured by a particular mathematical concept of computability. I present an analysis of calculability that is embedded in a rich historical and philosophical context, leads to precise concepts, but dispenses with theses.

To investigate effective calculability is to analyze symbolic processes that can in principle be carried out by calculators. This is a philosophical lesson we owe to Turing. Drawing on that lesson and recasting work of Gandy, I formulate boundedness and locality conditions for two types of calculators, namely, human computing agents and mechanical computing devices (discrete machines). The distinctive feature of the latter is that they can carry out parallel computations.

The analysis leads to axioms for discrete dynamical systems (representing human and machine computations) and allows the reduction of models of these axioms to Turing machines. Cellular automata and a variety of artificial neural nets can be shown to satisfy the axioms for machine computations.*

0. Background

The subtitle of this essay promises axioms for computability. Such axioms emerge from a *conceptual analysis* that begins with a straightforward observation: whatever we consider to be computable must be associated with computations that are carried out by some device or other. Consequently, we have to pay close attention to the nature of the device at hand, when thinking through the characteristic features that determine (the extension of) its notion of computability. My analysis builds on work by Turing and Gandy concerning computations that are carried out by human calculators and discrete machines, respectively.

I sharpen the informal concepts of computation for these two devices, specify rigorously their characteristic features, and formulate a representation theorem for the resulting systems of axioms. A broad methodological point can be immediately inferred: theses in the standard Church-Turing form are not needed to connect rigorously defined notions of computability with informally grasped concepts. It is however crucial to gain a proper understanding of the canonized connection between these notions, because the significance of logical results like Gödel's incompleteness theorems depends on it, as does the centrality of related issues in the philosophy of mind.

Part 1 articulates three principal *Church canons* supporting the thesis. For the canonical argument from confluence I distinguish between support that

derives from examining the effective calculability of number theoretic functions and support that is obtained through analyzing mechanical operations on symbolic configurations. The analysis of such operations, when carried out by a human computer, leads to Turing's claims in 1936. The arguments for these claims exploit *boundedness and locality conditions* that are presented in *Part 2*. Against this background I introduce in *Part 3* axioms for *Turing computers* and *Gandy machines*, list models, and formulate a representation theorem. That completes the conceptual analysis. I will conclude with remarks on Gödel, Turing, and philosophical errors.

1. Church canons¹

In a sense, we have to untangle the relation between the concept of computability and the concept of computability, understanding the first concept as *informally grasped* and the second as *rigorously defined*. If one takes Gödel's notion of general recursiveness as the rigorously defined concept and effective calculability as the informally grasped one, then *Church's Thesis* expresses the relation between *this* and *that* concept of computability for number-theoretic functions: they are co-extensional. To provide a proper perspective for the broader investigation, I will examine the early history of computability hinted at in these remarks.

1.1 The thesis. Gödel introduced general recursiveness for number theoretic functions in his 1934 Princeton Lectures via his equational calculus; he viewed it as a heuristic principle that the informal concept of *finite computation* can be captured by suitably general *recursions*. Refining and generalizing a notion of finitistically calculable functions due to Herbrand, Gödel defined a number theoretic function to be general recursive just in case it satisfies certain recursion equations and its values can be determined from the equations by simple steps, namely, replacement of variables by numerals and substitution of complex

¹According to the fifth edition of the Shorter OED, *canon* does not cover just ecclesiastical laws and decrees, but has also the meaning of "a general law, rule, or edict; a fundamental principle" since the late middle ages, and that of "a standard of judgement; a criterion" since the early 17th century.

closed terms by their numerical values. When he gave this definition in 1934 Gödel was not convinced, however, that the underlying precise concept of recursion was the most general one, and he expressed his doubts in conversation with Church. Nevertheless, Church formulated the thesis a year later for the first time in print. Here is the classical statement found in the abstract for Church's talk to the American Mathematical Society in December 1935:

... Gödel has proposed ... a definition of the term *recursive function*, in a very general sense. In this paper a definition of *recursive function of positive integers* which is essentially Gödel's is adopted. And it is maintained that the notion of an effectively calculable function of positive integers should be identified with that of a recursive function, since other plausible definitions of effective calculability turn out to yield notions that are either equivalent to or weaker than recursiveness.

Between Church's conversations with Gödel in 1934 and the formulation of the above abstract in 1935 some crucial developments had taken place in Princeton. Kleene and Rosser had done significant quasi-empirical work, convincing themselves and Church that all known effective procedures are λ -definable. Kleene had discovered his normal-form theorem and established the equivalence of Gödel's general recursiveness with μ -recursiveness. Finally, Church and Kleene had proved the equivalence of λ -definability and general recursiveness. All these developments are alluded to in Church's abstract, and they are interpreted as supporting the thesis, which was then, and is still now, principally defended on two grounds. *First* there is the quasi-empirical reason: all known calculable functions are general recursive. This point, though important, is clearly not decisive and will be taken up in the broader context of section 2.3. *Second*, there is the argument from confluence: a variety of mathematical computability notions all turn out to be equivalent. This second important point is however only convincing, if the "confluent" notions are of a quite different character and if there are independent reasons for believing that they capture the informal concept. Both Church and Gödel tried to give such independent reasons in 1936. Let me sketch their considerations.

1.2 Semi-circles. Church and Gödel took the evaluation of a function *in some form of the equational calculus* as the starting point for explicating the effective calculability of number theoretic functions. Church generalized broadly: an

evaluation is done in a logical calculus through a step-by-step process, and the steps must be elementary. Functions whose values can be computed in this way are, Church argued, general recursive. Gödel made a penetrating observation without giving an argument: the rules of the equational calculus are part of any adequate formal system of arithmetic, and the class of calculable functions is not enlarged beyond the general recursive ones, if the formal system is strengthened. This *absoluteness* of the notion was pointed out in a Postscriptum to 1936 for transfinite extensions of type theory and in the Princeton Bicentennial lecture ten years later for extensions of formal set theory. Gödel formulated the significance of his observation in the lecture as follows:

Tarski has stressed ... the great importance of the concept of general recursiveness (or Turing computability). It seems to me that this importance is largely due to the fact that with this concept 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. (Gödel 1946, p. 150)

But what is the argument for Church's claim, and what could it be for Gödel's? If one uses the strategic considerations underlying the proof of Kleene's normal-form theorem, it is in both cases easily established that the functions calculable in the broader frameworks are general recursive, as long as the steps in the logical systems are elementary, formal, ... well, general recursive. Church turned the elementary steps explicitly into general recursive ones, whereas Gödel could not but exploit the formal character of the theories at hand through their recursive presentation.

Taken as principled arguments for the thesis, Gödel's and Church's considerations rely on a hidden and semi-circular condition for steps. Hilbert and Bernays moved this step-condition into the foreground when investigating calculations in deductive formalisms and reckonable functions (*regelrecht auswertbare Funktionen*). They imposed explicitly *recursiveness conditions* on deductive formalisms and showed that formalisms satisfying these conditions have as their calculable functions exactly the general recursive ones. In this way they provided mathematical underpinnings for Gödel's absoluteness claim and for Church's argument, but only *relative* to the recursiveness conditions: the

crucial one requires the proof predicate of deductive formalisms, and thus the steps in formal calculations, to be primitive recursive!²

The work of Gödel, Church, Kleene and Hilbert & Bernays had intimate historical connections and is still of deep interest. It explicated calculability of functions by *exactly one core notion*, namely, calculability of their values in logical calculi via (a finite number of) elementary steps. But no one gave convincing and non-circular reasons for the proposed rigorous restrictions on the steps that are permitted in calculations. The question is, whether this stumbling block for a deeper analysis can be overcome. The answer lies in a motivated, general formulation of *constraints* on steps.

1.3 Symbolic processes. Church reviewed in 1937 the two classical papers by Turing and Post, which had been published in 1936. When comparing Turing computability, general recursiveness, and λ -definability he claimed “the first [of these notions] has the advantage of making the identification with effectiveness in the ordinary (not explicitly defined) sense evident immediately...” After all, Church reasoned, “To define effectiveness as computability by an arbitrary machine, subject to restrictions of finiteness, would seem to be an adequate representation of the ordinary notion, ...” The finiteness restrictions require that machines occupy only a finite space and that their working parts have finite size. Turing machines are obtained from such finite machines by further “convenient restrictions,” but “these are of such a nature as obviously to cause no loss of generality”. Church then observed, completely reversing Turing’s sequence of analytic steps, “a human calculator, provided with pencil and paper and explicit instructions, can be regarded as a kind of Turing machine”. He was obviously captured by the machine image and saw in it the reason for the deep interest of Turing’s computability notion. In sum, we have arrived at three *Church canons* in support of the thesis, namely, (i) the confluence of notions, (ii) the step-by-recursive-step argument, and (iii) the immediate evidence of the adequacy of Turing’s notion.

² These investigations are carried out in the second supplement of their *Grundlagen der Mathematik*, volume II.

In his reviews Church failed to recognize two crucial aspects of a dramatic shift in perspective. One aspect underlies the work of both Turing and Post, whereas the other is distinctively Turing's. The *first* aspect becomes visible when Turing and Post, instead of considering schemes for computing the values of number theoretic functions, look at identical symbolic processes that serve as building blocks for calculations. In order to specify such processes Post uses a human worker who operates in a symbol space and carries out, over a two-letter alphabet, exactly the kind of operations a Turing machine can perform. Post expects that his formulation will turn out to be equivalent to the Gödel-Church development. Given Turing's proof of the equivalence of his computability notion with λ -definability, Post's formulation is indeed equivalent.

Post asserts that "Church's identification of effective calculability with recursiveness" should be viewed as a "working hypothesis" in need of "continual verification". In sharp contrast, Turing attempts to give an analytic argument for the claim that these simple processes are sufficient to capture all human mechanical calculations. Turing exploits for his reductive argument broad constraints that are grounded in limitations of relevant capacities of the human computing agent. This is the *second* aspect of the novel perspective that made for genuine progress, and it is unique to Turing's work.

2. Computers

It is ironic that Post when proposing his worker model at no place used the fact that a human worker does the computing, whereas Turing who seems to emphasize machine computations examined explicitly *human* computations. Call a human computing agent who proceeds mechanically a *computer*; such a computer operates on finite configurations of symbols and, for Turing, deterministically so. The computer hovering about in Turing's paper is such a computer; computers in our contemporary sense are always called machines. Wittgenstein appropriately observed about Turing's machines that *these machines*

are humans who calculate.³ But how do we step from the calculations of computers to Turing machine computations?

2.1 Preliminary step. When Turing explores the extent of the computable numbers (or, equivalently, of the effectively calculable functions), he starts out by considering two-dimensional calculations “in a child’s arithmetic book”. Such calculations are first reduced to computations of *string machines*, and the latter are then shown to be equivalent to computations of a *letter machine*. Letter machines are ordinary Turing machines operating on one letter at a time, whereas string machines operate on finite sequences of letters. In the course of his reductive argument Turing formulates and uses broadly motivated constraints. The argument concludes as follows: “We may now construct a machine to do the work of the computer [computer in our terminology]. ... The machines just described [string machines] do not differ very essentially from computing machines as defined in § 2 [letter machines], and corresponding to any machine of this type a computing machine can be constructed to compute the same sequence, that is to say the sequence computed by the computer.” (Turing 1936, pp. 137-8)

For the presentation of Turing’s argument it is best to consider the description of Turing machines as Post production systems. This is most appropriate for a number of reasons. Post introduced this description in 1947 to establish that the word-problem of certain Thue-systems is unsolvable. Turing adopted it in 1950 when extending Post’s results, but also in 1954 when writing a wonderfully informative and informal essay on solvable and unsolvable problems. In addition, this description reflects directly the move in Turing’s 1936 to eliminate states of mind for computers⁴ in favor of “more physical counterparts”. Finally and most importantly, it makes perfectly clear that Turing

³ It is exactly right for Turing to look at *human* computations given the intellectual context that reaches back to at least Leibniz: the *Entscheidungsproblem* in the title of his 1936 paper asked for a procedure that can be carried out by humans; the restrictive formal conditions on axiomatic theories were imposed in mathematical logic to ensure intersubjectivity for humans, on a minimal cognitive basis.

⁴ Turing attributes states of mind only to *human computers*; machines have corresponding “m-configurations”.

is dealing with general symbolic processes, whereas the restricted machine model that results from his analysis almost obscures that fact.

2.2 Boundedness and locality. The constraints Turing imposes on symbolic processes derive from his central goal of isolating the most basic steps of computations, that is, steps that need not be further subdivided. This objective leads to the normative demand that the configurations, which are directly operated on, must be *immediately recognizable* by the computer. This demand and the evident limitation of the computer's sensory apparatus motivate most convincingly two central restrictive conditions:

(B) (*Boundedness*) A computer can immediately recognize only a bounded number of configurations.

(L) (*Locality*) A computer can change only immediately recognizable configurations.⁵

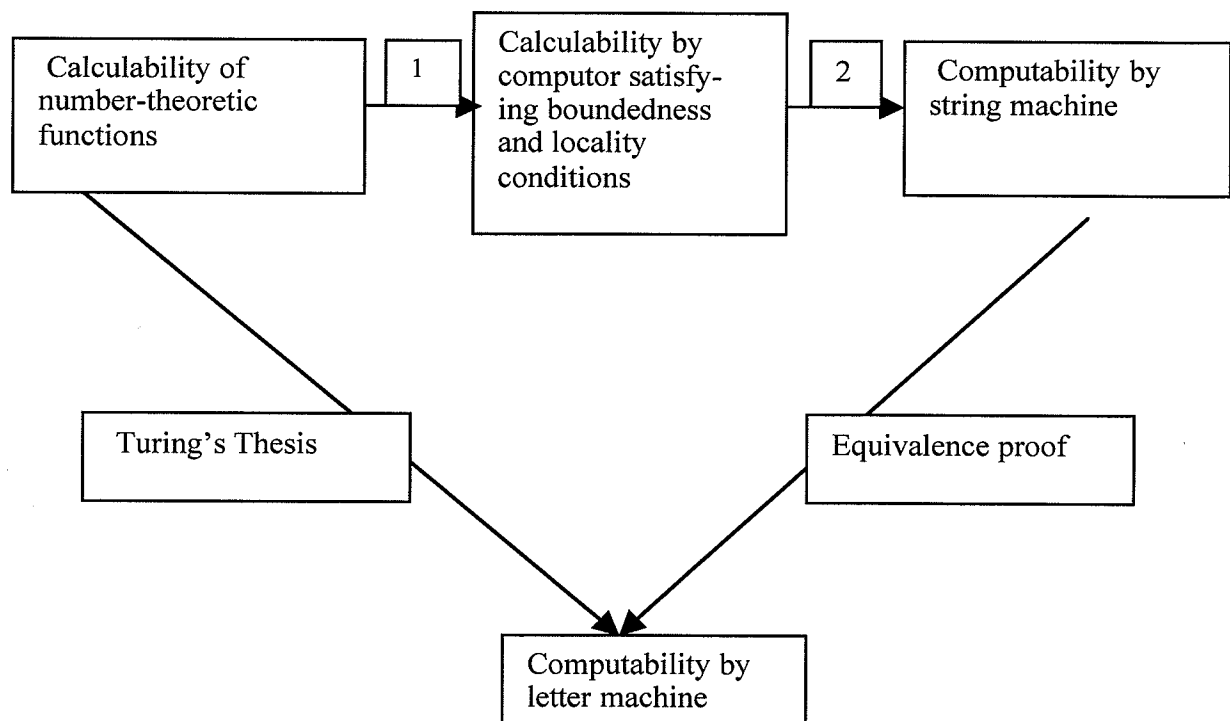


Diagram 1

⁵ The boundedness and locality conditions are violated in Gödel's equational calculus: the replacement operations naturally involve terms of arbitrary complexity. I.e., the shift from arithmetic calculations to symbolic processes is absolutely crucial.

Turing's considerations leading from operations of a computer on a two-dimensional piece of paper to operations of a letter machine on a linear tape are represented schematically in *diagram 1*: Step 1 indicates Turing's analysis, whereas 2 refers to Turing's *central thesis* asserting that the calculations of a computer can be carried out by a string machine.

This remarkable progress has been achieved by bringing in, crucially and correctly, the computing agent who carries out the mechanical processes. Yet Turing finds the argument mathematically unsatisfactory as it involves an appeal to *intuition* in support of the central thesis, i.e., the ability of "making spontaneous judgments, which are not the result of conscious trains of reasoning". (Turing 1939, pp. 208-9) What more can be done?

2.3 Generalizations. At least two kinds of inductive support can be given for the quasi-empirical claim that all known effective procedures are general recursive or Turing computable. Turing provided in his paper one kind, by showing that large classes of numbers are indeed machine computable; Post suggested providing in his 1936 a second kind, by reducing ever-wider formulations of combinatory processes (as production systems) to his worker model.⁶ This inductive support can be strengthened further through considering more general symbolic configurations with associated complex substitution operations.⁷ In the spirit of this approach we can ask with Post, when have we gathered sufficient support to view the thesis as a *natural law*?

Gödel and Church faced in their analysis of effective calculability the stumbling block of having to define the elementary character of steps, rigorously and without semi-circles. Turing and Post faced at this point, it seems, a problem akin to that of induction. However, their fundamental difficulties are really the same and can be pinpointed more relevantly and quite clearly, as they are related to the looseness of the above restrictive conditions and the corresponding vagueness of the central thesis. These difficulties would be

⁶ Post of course did provide such reductions in his 1943 whose origins go back to investigations in the very early 1920s; see note 18 of Post's paper.

⁷ In Sieg and Byrnes 1996 that is done for K-graphs and K-graph machines; this is a generalization of the work on algorithms by Kolmogorov and Uspensky.

addressed by answering the questions, What are symbolic configurations? What changes can mechanical operations effect? – Even without giving rigorous answers, some well-motivated ideas can be formulated for computers: (i) they operate deterministically on finite configurations; (ii) they recognize only a bounded number of different kinds of patterns (in these configurations); (iii) they operate locally on exactly one of the patterns⁸; (iv) they assemble the next configuration from the original one and the result of the local operation. Exploiting these ideas I will attack the problem with a familiar tool, the axiomatic method.

However, before formulating the axioms for Turing computers, I discuss yet another sense of “generalization” that is relevant here. Gandy proposed in his 1980 a characterization of *machines* or, more precisely, *discrete mechanical devices*. The latter clause was to exclude analogue machines from consideration. The novel aspect of Gandy’s proposal was the fact that it incorporated parallelism in perfect generality. Gandy used, as Turing did, a *central thesis*: any discrete mechanical device satisfying some informal restrictive conditions can be represented as a particular kind of dynamical system. Instead, I characterize a *Gandy machine* axiomatically based on the following idea: the machine has to recognize *all* the patterns (from a bounded set) in a given finite configuration, act on them locally in parallel, and assemble the results of these local computations into the next configuration. As in the case of Turing computers, the configurations are finite, but unbounded; the generalization is simply this: there is no fixed bound on the number of patterns that such configurations may contain. To help the imagination a bit, the reader should think of the Post-presentation of a Turing machine and the Game of Life as typical examples of a Turing computer and Gandy machine, respectively.

⁸ Every finite configuration contains exactly one of the patterns.

3. Axiomatics⁹

The axioms are formulated for discrete dynamical systems and capture the above general ideas precisely; they should be viewed as determining classes of “algebraic structures” of which particular models of computation are instantiations. In the first subsection the general mathematical set-up for the axioms is discussed, whereas the specific principles for *Turing computers* and *Gandy machines* are formulated in the second subsection. The axioms for Turing computers are motivated by the restrictive conditions for human computing. The axioms for Gandy machines are to capture the characteristic features of finite machines (performing parallel computations). The restrictive conditions are motivated by purely physical considerations: the uncertainty principle of quantum mechanics justifies a lower bound on the size of distinguishable “atomic” components, and the theory of special relativity yields an upper bound on signal propagation. Together, these conditions justify boundedness and locality conditions for machines in the very way sensory limitations do for computers.

3.1 Patterns & local operations. We consider pairs $\langle D, F \rangle$ where D is a *class of states* and F an *operation* from D to D transforming a given state into the next one. States are finite objects and are represented by non-empty hereditarily finite sets over an infinite set of atoms. Such sets reflect states of computing devices just as other mathematical structures represent states of nature. Obviously, any ϵ -isomorphic set can replace a given one in this reflective role, and so we consider *structural classes* D , i.e., classes of states that are closed under ϵ -isomorphisms. What invariance properties should the state transforming operations F have, i.e., how should the F -images of ϵ -isomorphic states be related? These and other structural issues will be addressed now.

For the general set-up we notice that any ϵ -isomorphism between states is an extension of some permutation π on atoms. Letting $\pi(x)$ stand for the result

⁹ I hope the overall structure of the considerations will be clear from this informal presentation; for mathematical details *Gandy 1980* and *Sieg 2002B* should be consulted.

of applying the \in -isomorphism determined by a permutation π to the state x , the requirement on F fixes the dependence of values on just structural features of a set, not the nature of its atoms: $F(\pi(x))$ is \in -isomorphic to $\pi(F(x))$, and this isomorphism must be the identity on the atoms occurring in $\pi(x)$; we say that $F(\pi(x))$ and $\pi(F(x))$ are \in -isomorphic over $\pi(x)$ and write $F(\pi(x)) \cong_{\pi(x)} \pi(F(x))$. Note that we do not require $F(\pi(x)) = \pi(F(x))$; that would be far too restrictive as new atoms may expand the state x , and it should not matter which new atoms are chosen. The requirement $F(\pi(x)) \cong \pi(F(x))$, on the other hand, would be too loose, as we want to guarantee the physical persistence of atomic components.

Now we turn to *patterns* and *local* operations. If x is a given state, regions of the next state are determined *locally* from particular *parts* for x on which the computer can operate.¹⁰ *Boundedness* requires that there are only finitely many different kinds of such parts, i.e., each part lies in one of a finite number of isomorphism types that are also called *stereotypes*. A maximal part y for x of a certain stereotype is a *causal neighborhood* for x , briefly $y \in Cn(x)$; we call the elements of $Cn(x)$ also *patterns*. Finally, the local change is effected by a structural operation G that works on unique causal neighborhoods. The values of G are in general not exactly what we need in order to assemble the next state, because the configurations may have to be expanded and that expansion involves the addition and coordination of new atoms. To address that issue we introduce *determined regions* $Dr(z, x)$ of a state z ; they are \in -isomorphic to $G(y)$ for some causal neighborhood y for x (and must satisfy a technical condition on the “newness” of atoms).

3.2 Axioms & a theorem. Recalling the boundedness and locality conditions for computers, we define $M = \langle S; T, G \rangle$ to be a *Turing Computer on S*, where S is a structural class, T a finite set of stereotypes, and G a structural operation on $\cup T$, if and only if, for every $x \in S$ there is a $z \in S$, such that

¹⁰ A connected subtree y of the \in -tree for x is called *part* for x , briefly $y <^* x$, if $y \neq x$ and y has the same root as x and its leaves are also leaves of x .

(LC.0) $(\exists!y) y \in \text{Cn}(x)$,

(LC.1) $(\exists!v \in \text{Dr}(z,x)) v \cong_x G(\text{cn}(x))$,

(GA.1) $z = (x \setminus \text{Cn}(x)) \cup \text{Dr}(z,x)$.

$(\exists!y)$ is the existential quantifier expressing uniqueness; in (LC.1), $\text{cn}(x)$ denotes the unique causal neighborhood guaranteed by (LC.0). (As in the case of Gandy Machines below, LC abbreviates local causation, whereas GA stands for global assembly.) – The state z is determined uniquely up to ϵ -isomorphism over x . A computation by \mathbf{M} is a finite sequence of transition steps involving \mathbf{G} that is halted when the operation on state z yields z as the next state. A function F is (Turing) *computable* if and only if there is a Turing computer \mathbf{M} from whose computation results one can determine – under a suitable encoding and decoding – the values of F for any of its arguments. A Turing machine is easily seen to be a Turing computer.

Generalizing these considerations to graph machines, for example, one notices quickly complications. When several new atoms are being introduced in the image of some causal neighborhood as well as in the next state, the new atoms have to be structurally coordinated. That can be achieved by a second local operation and a second set of stereotypes. Causal neighborhoods of type 1 are parts of larger neighborhoods of type 2 and the overlapping determined regions of type 1 must be parts of determined regions of type 2, so that they fit together appropriately. (Determined regions “overlap”, if the intersection of their sets of new atoms is non-empty.)

For machines that carry out parallel computations, we thus need in addition to the finitely many stereotypes and the structural operation working on them a second set of stereotypes together with a second structural operation, which allow the machine to assemble the determined regions. This is reflected by separating the principles for Gandy machines into two kinds (as we did for Turing computers), those of *Local Causation* (LC) and those of *Global Assembly* (GA): $\mathbf{M} = \langle \mathbf{S}; \mathbf{T}_1, \mathbf{G}_1, \mathbf{T}_2, \mathbf{G}_2 \rangle$ is a *Gandy machine on S*, where \mathbf{S} is a structural

class, T_i a finite set of stereotypes, G_i a structural operation on $\cup T_i$, if and only if, for every $x \in S$ there is a $z \in S$, such that

(LC.1): $(\forall y \in Cn_1(x)) (\exists! v \in Dr_1(z, x)) v \cong_x G_1(y)$;

(LC.2): $(\forall y \in Cn_2(x)) (\exists v \in Dr_2(z, x)) v \cong_x G_2(y)$;

(GA.1): $(\forall C) [C \subseteq Dr_1(z, x)) \& \cap \{Sup(v) \cap A(z, x) \mid v \in C\} \neq \emptyset \rightarrow$
 $(\exists w \in Dr_2(z, x)) (\forall v \in C) v <^* w]$;

(GA.2): $z = \cup Dr_1(z, x)$.

$A(z, x)$ consists of the new atoms that have been introduced into z , i.e., $A(z, x) = Sup(z) \setminus Sup(x)$. Thus, the condition $\cap \{Sup(v) \cap A(z, x) \mid v \in C\} \neq \emptyset$ in (GA.1) expresses that the determined regions v in C have common new atoms, i.e., they overlap. The restrictions for Gandy machines, as those for Turing computers, amount to boundedness and locality conditions. They are justified *directly* by two physical limitations, namely, a lower bound on the size of atoms and an upper bound on the speed of signal propagation. With these remarks I actually completed the foundational work and can describe now some important mathematical facts for Gandy machines.

The central facts are these: (i) the state z following x is determined uniquely up to ϵ -isomorphism over x , and (ii) Turing machines can effect such transitions. The proof of the first fact contains the combinatorial heart of matters and uses crucially the first global assembly condition. The proof of the second fact is rather direct. Only finitely many finite objects are involved in the transition, and all the axiomatic conditions are decidable. Thus, a search will allow us to find z . This can be understood as a *Representation Theorem*: any particular Gandy machine is computationally equivalent to a two-letter Turing machine, as Turing machines are also Gandy machines. Indeed, there is a rich variety of additional models, as the game of life, other cellular automata, and many artificial neural nets are Gandy machines. (Cf. DiPisapia 2000.)

4. Adequacy & philosophical errors

So what? What have we gained? In very broad terms, taken from Hilbert, we have gained *eine Tieferlegung der Fundamente* (a deepening of the foundations) via the axiomatic method. In a conversation with Church in early 1934, Gödel found Church's proposal to identify effective calculability with λ -definability "thoroughly unsatisfactory". As a counter-proposal he suggested "to state a set of axioms which would embody the generally accepted properties of this notion [i.e., effective calculability], and to do something on that basis". Perhaps, the remarks in the 1964 Postscriptum to the Princeton Lectures of 1934 echo those earlier considerations. "Turing's work gives," according to Gödel, "an analysis of the concept of 'mechanical procedure' This concept is shown to be equivalent with that of a 'Turing machine'." Gödel did neither elucidate these remarks, nor did he articulate, what the generally accepted properties of effective calculability might be or what might be done on the basis of an appropriate set of axioms.

The work on which I reported substantiates Gödel's remarks in the following sense: it formulates axioms for the concept "mechanical procedure" and it shows that this axiomatically characterized concept is computationally equivalent to that of a Turing machine. Indeed, it does so for two such concepts, namely when the computing agents are computers, respectively discrete machines. These considerations use only "generally accepted properties" of the informal concepts and avoid any appeal to theses, whether central or not. As to the correctness of the underlying analyses, an appeal to some understanding can no more be avoided in this case than in any other case of an axiomatically characterized (class of) mathematical structure(s) intended to mirror broad aspects of physical or intellectual reality. The general point is this: we don't have to face anything mysterious surrounding the concept of calculability; rather, we have to face the ordinary issues for the adequacy of mathematical concepts, and these are of course non-trivial!¹¹ From a slightly different and complementary perspective, the function of the axiom systems for computing devices can be seen as being similar to that of the axiom systems for the classical algebraic

¹¹ Other examples of such analyses are provided by Dedekind's work on continuous domains (the reals) and simply infinite systems (natural numbers).

structures like groups, rings or fields, namely, to abstract the essential aspects from a wide variety of instances and point to deep structural analogies; they explain here, by way of the representation theorem, the computational equivalence of their models.

In the central case under discussion, Turing computability, its adequacy is still fraught with controversy and often misunderstanding. The controversy begins with the very question, what the intended informal concept is. For example, Gödel spotted in 1972 a “philosophical error” in Turing’s work, *assuming* that Turing’s argument in the 1936 paper was to show that “mental procedures cannot go beyond mechanical procedures”. He considered the argument as inconclusive. Indeed, Turing does not give a conclusive argument for Gödel’s claim, but it has to be added that he did not intend to argue for it. Even in his work of the late 1940’s and early 1950’s that deals explicitly with mental processes, Turing does not argue, “mental procedures cannot go beyond mechanical procedures”.

Mechanical processes are, in this later work, still made precise as Turing machine computations; machines that might exhibit intelligence have in contrast a more complex structure than Turing machines. Conceptual idealization and empirical adequacy are being sought for quite different purposes, and Turing is trying to capture clearly what Gödel found missing in the would-be analysis of a broad concept of humanly effective calculability, namely, “... that mind, in its use, is not static, but constantly developing”. The real difference between Turing’s and Gödel’s views, it seems, is Gödel’s belief that it is “a prejudice of our time” that “[t]here is no mind separate from matter”. This is reported by Wang. Gödel expected, also according to Wang, that this prejudice “will be disproved scientifically (perhaps by the fact that there aren’t enough nerve cells to perform the observable operations of the mind)”. Clearly, Turing did not share these expectations.

There are many fascinating issues concerning physical and mental processes that may or may not have adequate computational models. They are empirical, conceptual, mathematical ... well, indeed, richly interdisciplinary.

Steps towards their clarification or resolution will be most illuminating. Why, let me ask, are we interested so deeply in computations? – One answer might be, we want to *determine* states from other states, be they mathematical, physical or mental; and we want to do that effectively and in a sharply intersubjective way that makes use of adequate symbolic representations.

References

- Church, A.
 1936 An unsolvable problem of elementary number theory; American Journal of Mathematics 58, 345-363; reprinted in *Davis* 1965.
 1937 Review of (Turing 1936); Journal of Symbolic Logic 2, 40-41.
- Davis, M.
 1965 (ed.), *The Undecidable*, Basic papers on undecidable propositions, unsolvable problems and computable functions; Raven Press, Hewlett, New York.
- De Pisapia, N.
 2000 *Gandy Machines: an abstract model of parallel computation for Turing Machines, the Game of Life, and Artificial Neural Networks*; M.S. Thesis, Carnegie Mellon University, Pittsburgh.
- Gandy, R.
 1980 Church's Thesis and principles for mechanisms; in: *The Kleene Symposium* (edited by J. Barwise, H.J. Keisler and K. Kunen, North-Holland, 123-148.
- Gödel, K.
 1934 On undecidable propositions of formal mathematical systems; in: *Collected Works I*, 346-371.
 1936 Über die Länge von Beweisen; in: *Collected Works I*, 396-399.
 1946 Remarks before the Princeton bicentennial conference on problems in mathematics; in: *Collected Works II*, 150-153.
 1986-
 2003 *Collected Works*, volumes I – V; Oxford University Press.
- Hilbert, D. and P. Bernays
 1939 *Die Grundlagen der Mathematik II*; Springer Verlag, Berlin.
- Kolmogorov, A.N. and V.A. Uspensky
 1958 On the definition of an algorithm; Uspekhi Mat. Nauk 13 (Russian), 1958; English translation in: *AMS Translations*, 2, 21 (1963), 217-245.
- Post, E.
 1936 Finite combinatory processes. Formulation I. Journal of Symbolic Logic 1, 103-5.
 1943 Formal reductions of the general combinatorial decision problem; American Journal of Mathematics, 65 (2), 197-215.
 1947 Recursive unsolvability of a problem of Thue; Journal of Symbolic Logic 12, 1-11.

Sieg, W.

- 1994 Mechanical procedures and mathematical experience, in: *Mathematics and Mind* (A. George, ed.), Oxford University Press, 71-117.
- 1997 Step by recursive step: Church's analysis of effective calculability, *Bulletin of Symbolic Logic* 3, 154-80.
- 2002A Calculations by man and machine: conceptual analysis; *Lecture Notes in Logic* 15, 390-409.
- 2002B Calculations by man and machine: mathematical presentation; in: *In the Scope of Logic, Methodology and Philosophy of Science*, volume one of the 11th International Congress of Logic, Methodology and Philosophy of Science, Cracow, August 1999 (P. Gärdenfors, J. Wolenski and K. Kijania-Placek, eds.), Synthese Library volume 315, Kluwer, 247-262.

Sieg, W. and J. Byrnes

- 1996 K-Graph machines: generalizing Turing's machines and arguments; in: *Gödel '96* (P. Hajek, ed.), *Lecture Notes in Logic* 6, Springer Verlag, 98-119.

Turing, A.

- 1936 On computable numbers, with an application to the Entscheidungsproblem; *Proc. London Math. Soc.*, series 2, 42, 230-265; reprinted in *Davis 1965*.
- 1939 Systems of logic based on ordinals; *Proc. London Math. Soc.*, series 2, 45, 161-228; reprinted in *Davis 1965*.
- 1950 The word problem in semi-groups with cancellation; *Ann. of Math.* 52, 491-505.
- 1954 Solvable and unsolvable problems; *Science News* 31, 7-23; reprinted in *Collected Works of A.M. Turing: Mechanical intelligence*, (D.C. Ince, ed.), North-Holland, 1992.

* This essay is based on two papers published in 2002, but whose methodological considerations I would like to bring out more distinctly. I presented versions of this essay under the title *Beyond Church Canons* in the Distinguished Lecture Series (Haverford College, October 2002), in the Annual Lecture Series at the Center for Philosophy of Science (University of Pittsburgh, January 2004), at the Colloquium of the IHPST (Sorbonne, May 2004), as well as at the Colloquium of the Department of Philosophy (University of Florence, November 2004). For detailed discussions of the origins and developments of computability, see also Sieg 1994, 1997 and the rich literature that is referred to.