# Paper Machines

by

Daniele Mundici and Wilfried Sieg

February 1994

Report CMU-PHIL-48


Philosophy
Methodology
Logic

Pittsburgh, Pennsylvania 15213-3890

# Paper Machines*

**Daniele Mundici and Wilfried Sieg**

*** An expanded version of this paper, *Mathematics studies machines,* is to appear (in Italian) in: Le Scienze della Mente, C. Mangione (ed.), a volume of the encyclopedia Le Scienze e le Tecnologie, ieri oggi e domani, Grandi Opere, Milan, Italy. -- P. Odifreddi read that paper very carefully; we profited from his suggestions also for the present essay.**

**PROLOGUE**. Machines were introduced as calculating devices to simulate operations carried out by human computors following fixed algorithms. This is true for the early mechanical calculators devised by Pascal and Leibniz, for the analytical engines built by Babbage, and for the theoretical machines introduced by Turing. The distinguishing feature of the latter is their universality: They are claimed to be able to capture any algorithm whatsoever and, conversely, any procedure they can carry out is evidently algorithmic. The study of such *paper machines* (by mathematical means) is the topic of our essay; section zero provides necessary *logical background*.

In section one, *Steps (towards Turing's analysis)*, we examine the analyses of effective calculability given in the thirties. We argue that Gödel, Church & Kleene, and Hilbert & Bernays focused on *one* central informal notion, namely the stepwise calculation by a human computor in a symbolic calculus. In each case a serious stumbling-block to a convincing analysis emerged: either only very specific, obviously mechanical steps were allowed or the restricted nature of steps in computations was characterized by recourse to (primitive) recursiveness. Turing overcame this critical difficulty by formulating axiomatic conditions for a computor and by reducing stepwise calculations (carried out by a computor that satisfies the conditions) to computations (carried out by a machine that can be described mathematically in a most convenient way).

In section two, *Steps (towards the mathematical theory)*, we describe results that are central to recursion theory and that reinforce the conceptual analysis given in section one. The idea of stepwise calculation is reflected directly in Kleene's Normal Form Theorem which implies, with hardly any additional work, the existence of a universal machine. Then we discuss the Halting Problem and a concept of effective reducibility: by reducing the Halting Problem to the decision problem Turing established the unsolvability of the latter. Turing's computability notion allowed Gödel to formulate the incompleteness theorems for *all* sufficiently strong "formal" theories, because that notion captured, in Gödel's judgement, the informal concept of calculability exactly.

In section three, *Physical limits*, we pursue the analysis of computation steps in a quite different way. The axiomatic conditions for computors were motivated by limitations of the human sensory apparatus, but Turing saw their ultimate justification as given by

limitations of human memory. Physics provides grounds for such memory limitations; we argue also that there are physical reasons, why steps cannot be accelerated arbitrarily and why they cannot be made arbitrarily complex. A different sense of the mathematical study of machines emerges, namely, the formulation of principles governing the operations of physically realizable machines.

**0 LOGICAL BACKGROUND.** The central issues can be traced back to Greek mathematics and philosophy, as they concern the axiomatic presentation of geometry (culminating in Euclid's *Elements*) and the formalization of logical reasoning (by Aristotle and, for sentential logic, by the Stoics). But only Frege provided in his *Begriffsschrift* an expressive formal language and a logical calculus that allowed him to realize the earlier intentions with respect to mathematics. He required that all assumptions of a proof be explicitly formulated in the language and that each step in a proof be taken in accord with one of the antecedently specified rules of the calculus. With this sharpening of the axiomatic method Frege analyzed the "epistemological nature" of theorems. That could be done, because the rules do not require contentual knowledge; indeed, Frege claimed that in his system "inference is conducted like a calculation".

**0.1 DECISION PROBLEM.** More than half a century later Gödel referred back to Frege when he described in his *1933* "the outstanding feature of the rules of inference" in a formal mathematical system. The rules, Gödel said, "refer only to the outward structure of the formulas, not to their meaning, so that they can be applied by someone who knew nothing about mathematics, or by a machine." Frege had not viewed the possibility of mechanically drawing inferences as one of the logically significant achievements of his *Begriffsschrift*. Hilbert, however, grasped the potential of this aspect, radicalized it, and exploited it in his formulation and pursuit of the consistency problem. His research program started in a rough and tentative way around 1900 and was pursued with great intensity in the twenties. The strict formalization of mathematics seemed to open up new ways of addressing foundational issues and of solving mathematical problems - through calculation.

The most famous problem among these is the *Entscheidungsproblem* or decision problem. It is closely related to the consistency problem and was pursued, for example by Herbrand, on account of this connection. Its classical formulation is found in Hilbert and

Ackermann's 1928 book *Grundzüge der Theoretischen Logik:* "The Entscheidungsproblem is solved if one knows a procedure that permits the decision concerning the validity, respectively, satisfiability of a given logical expression by a finite number of operations." (The problem had been formulated by Hilbert already in lectures given in Göttingen during the winter term 1917/18.)

Hilbert and Ackermann emphasized the fundamental importance of the decision problem, and researchers in the Hilbert school realized that a positive solution would allow the decision concerning the provability of any mathematical statement (within finitely axiomatized theories), von Neumann *1927* took that as sufficient reason to expect a negative solution, but claimed that there was no clue as to how an actual proof of undecidability would go. To support this claim he pointed to the deep conceptual problem. There were proofs for the unsolvability of well-known mathematical problems; however, all such impossibility results were given relative to a determinate class of admissible means (e.g., doubling the cube by using only ruler and compass). And von Neumann saw exactly here the problem; any negative solution to the Entscheidungsproblem would require a mathematically precise answer to the question: "What are mechanical procedures?"

Mechanical procedures for natural numbers were familiar from mathematical practice: The values of all "primitive recursive" functions can be calculated mechanically. And most functions from elementary number theory are in this class, as was established already by Skolem *1923]* for example, addition, multiplication, exponentiation, the factorial function n!, the sequence of prime numbers, and the sequence of Fibonacci numbers are all primitive recursive. However, Ackermann had discovered in the mid-twenties a non-primitive recursive function whose values could be determined by a mechanical procedure.[1]

**0.2 INCOMPLETENESS.** In his *1931* paper Gödel used primitive recursive functions to describe, after an effective number theoretic coding, the syntax of particular "formal" theories. Since Gödel strove to use a general concept of formality through the underlying concept of calculability, there was no reason to focus attention on theories whose syntax could be presented primitive recursively. And

---

Schemes for non-primitive recursive functions whose values are calculable were investigated systematically by Peter in the mid-thirties; see her book *Rekursive Funktionen.*

in the 1934 Princeton Lectures Gödel tried very hard to make his incompleteness results less dependent on particular formalisms. But he did not succeed in resolving the main conceptual issue to his own satisfaction, i.e., to give a general notion of "formal theory". He viewed the primitive recursive definability of formulas and proofs as a precise condition which in *practice* suffices to describe formal systems, but he was searching for a condition that would suffice in *principle*. In what direction could one search? Gödel considered it as an "important property" that the value of a primitive recursive function can be computed by a "finite procedure", for any argument; he added in a footnote:

> The converse seems to be true if, besides recursions according to the scheme (2) [of primitive recursion], recursions of other forms ... are admitted. This cannot be proved, since the notion of finite computation is not defined, but it can serve as a heuristic principle.

In the last section of his Princeton lectures Gödel defined "general recursive functions". They are obtained as unique solutions of functional equations expressing more than primitive recursions, and - that was crucial for Gödel - their values are computed in a calculus with particular mechanical rules.

**0.3 THESIS**. The footnote we quoted in the last paragraph may seem to express a form of Church's Thesis; but Gödel emphasized in a 1965 letter to Martin Davis that no formulation of that thesis was intended: "The conjecture stated there only refers to the equivalence of 'finite (computation) procedure' and 'recursive procedure'. However, I was, at the time of these lectures, not at all convinced that my concept of recursion comprises all possible recursions; ... " At the time, Gödel was equally unconvinced by Church's proposal to identify effective calculability with $\lambda$-definability. Church recalled in a letter to Kleene (dated November 29, 1935) a conversation with Gödel in early 1934, when Gödel called his proposal "thoroughly unsatisfactory". Nevertheless, Church announced his thesis in a talk at the meeting of the American Mathematical Society on April 19, 1935; he formulated it in terms of recursiveness, not $\lambda$-definability. In his subsequent 1936 paper *An unsolvable problem of elementary number theory* Church wrote:

> The purpose of the present paper is to propose a definition of effective calculability which is thought to correspond satisfactorily to the somewhat vague intuitive notion in terms of which problems of this class are often stated, and to show, by means of an example, that not every problem of this class is solvable.

Church proposed again to identify effective calculability with recursiveness. The fact that A,-definability was known to be an equivalent concept simply added for Church "... to the strength of the reasons adduced below for believing that they [these precise concepts] constitute as general a characterization of this notion [i.e., effective calculability] as is consistent with the usual intuitive understanding of it.ff These reasons will be analyzed in section 1.0.

**1 STEPS (towards Turing's Thesis).** According to the conventional view, the work of Gödel, Church, Turing, and others (e.g., Kleene, Post, Hilbert, Bernays) provided mathematical definitions of mechanical procedures. The fact that these definitions turned out to be equivalent, in the sense of characterizing the same class of number theoretic functions as computable, has been taken to support Church's Thesis. The question for us is: What are the grounds for accepting the various notions as constituting a mathematical description of mechanical procedures? The considerations in this section are based on [Sieg 1994], as are those of the preceding section; good complementary discussions are found in [Gandy 1988], [Tamburrini], and [Odifreddi, part I, section 8].

**1.0 STEP-BY-STEP.** In his *1936* Church pointed out that the notion *calculability in a logic* suggests itself as a way to characterize effective calculability of number theoretic functions, and he argued that it does not lead to a definition more general than recursiveness. Let us indicate the argument: Church considered a logic L whose language contains the equality symbol =, a symbol { }( ) for the application of a unary function symbol to its argument, and numerals for the positive integers. He called unary functions F *effectively calculable* if and only if there is an expression f in L such that {f}(M)=N is a theorem of L exactly when F(m)=n; M and N are expressions of L that stand for the positive integers m and n. Church claimed that such F are recursive, when L satisfies conditions that guarantee essentially the recursive enumerability of $L^f$s theorem predicate; the claim follows by an unbounded search. The crucial condition in Church's list requires the steps in derivations of equations to be, well, recursive! Here we hit a serious stumbling-block for Church's analysis, since an appeal to the thesis when arguing for it is logically circular. And yet, Church's argument achieves something: The general concept of calculability is explicated as derivability in a logic, and the step-condition is used to sharpen the idea that we operate by effective rules in such a

formalism. We suggest that the claim "Steps of any effective procedure are recursive!" be called **Church's Central Thesis.**

ChurcIVs concept *calculability in a logic* is extremely natural and fruitful; it is directly related to decidability (Entscheidungs-definitheit) for relations and classes in Gödel's *1931* and to repre-sentability in his Princeton lectures; Gödel defined the very notion in *1936* and emphasized its type-absoluteness. Finally, in *1946* Gödel took absoluteness in a general sense as the main reason for the special importance of recursiveness: Here we have the first interesting epistemological notion whose definition is not dependent on the chosen formalism. But the stumbling-block Church had to face shows up also here, as absoluteness is achieved only relative to formal systems.

The definition of absoluteness Gödel gave in 1946 is explicit already in Hilbert and Bernays[1] *Grundlagen der Mathematik II.* They called a number-theoretic function *reckonable according to rules* (regelrecht auswertbar), if it is calculable in some deductive for-malism, and formulated three recursiveness conditions for such formalisms. Then they proved: (i) a function that is calculable in some deductive formalism satisfying the recursiveness conditions can be computed in a very restricted number theoretic formalism, and (ii) the functions calculable in the latter formalism are exactly the recursive functions.

Hilbert and Bernays[1] analysis is a most satisfactory capping of the development from Entscheidungsdefinitheit to an "absolute" no-tion of computability. But their analysis does not overcome the major stumbling-block; rather, it puts the stumbling-block in plain view through the recursiveness conditions that deductive formalisms must satisfy. The crucial condition requires the proof predicate for such formalisms to be primitive recursive! We want to show now, how Turing got around this fundamental difficulty and start out by describing Turing machines; in the presentation of these machines we follow Davis *1958,* not Turing's original paper.

**1.1 MACHINES & WORKERS.** A Turing machine consists of a fi-nite, but potentially infinite tape; the tape is divided into squares, and each square may carry a symbol from a finite alphabet, say, just the two-letter alphabet consisting of 0 and 1. The machine is able to scan one square at a time and perform, depending on the content of the observed square and its own internal state, one of four op-

erations: print 0, print 1, or shift attention to one of the two immediately adjacent squares. The operation of the machine is given by a finite list of commands in the form of quadruples qjS|<Ciq$_m$ that express: if the machine is in internal state *q*\ and finds symbol Sk on the square it is scanning, then it is to carry out operation Q and change its state to $q_m$. The deterministic character of the machine operation is guaranteed by the requirement that a program must not contain two different quadruples with the same first two components.

In 1936, the very year in which Turing's paper appeared, Post published a strikingly similar computation model, where a worker operates in a *symbol space* consisting of "a two way infinite sequence of spaces or boxes ... fi".[2] The worker can be in and operate in just one box at a time; the boxes admit only two conditions: they can be empty or unmarked, or they can be marked by a single sign, say a vertical stroke. The worker can perform a number of *primitive acts',* namely, make a vertical stroke [V], erase a vertical stroke [E], move to the box immediately to the right [$M_r$] or to the left [M|] (of the box he is in), and determine whether the box he is in is marked or not [D]. In carrying out a particular combinatory process the worker begins in a special box and then follows directions from a finite, numbered sequence of instructions. The i-th direction, i between 1 and n, is in one of the following forms: (i) carry out act V, E, $M_r$, or M| and then follow direction jj, (ii) carry out act D and then, depending on whether the answer is positive or negative, follow direction jj[1] or jj". (Post has a special stop instruction, but that can be replaced by the convention to halt the process, when the number of the next direction is greater than n.)

Are there intrinsic reasons for choosing Formulation 1, except for its simplicity and Post's expectation that it will turn out to be equivalent to recursiveness? An answer to this question is not clear from Post's paper, at the very end of which he wrote:

> The writer expects the present formulation to turn out to be equivalent to recursiveness in the sense of the Gödel-Church development. Its purpose, however, is not only to present a system of a certain logical potency but also, in its restricted field, of psychological fidelity. In the latter sense wider and wider formulations are contemplated. On the other hand, our aim will be to show that all such are logically reducible to formulation 1. We offer this conclusion at the present moment as a *working hypothesis.* And to our mind such is Church's identification of effective calculability with recursiveness.

---

[2] Post remarks that the infinite sequence of boxes is ordinally similar to the series of integers and can be replaced by a potentially infinite one, expanding the finite sequence as necessary.

Investigating wider and wider formulations and *reducing* them to *Formulation 1* would change for Post this "hypothesis not so much to a definition or to an axiom but to a *natural law*" . It is methodologically remarkable that Turing proceeded in *exactly* the opposite way when trying to justify that all computable numbers are machine computable or, in our way of speaking, that all effectively calculable functions are Turing computable: He did not try to extend a narrow notion reducibly and obtain in this way quasi-empirical support, but analyzed the intended broad concept and reduced it to a narrow one - once and for all.

**1.2 CONCEPTUAL ANALYSIS.** Turing's *On computable numbers* opens with a description of what is ostensibly its subject, namely, "real numbers whose expressions as a decimal are calculable by finite means". Turing is quick to point out that the problem of explicating "calculable by finite means" is the same when considering, e.g., computable functions of an integral variable. Thus it suffices to address the question: "What does it mean for a real number to be calculable by finite means?" In §9 he argues that the operations of his machines "include all those which are used in the computation of a number". But he does not try to establish the claim directly; he rather attempts to answer what he views as "the real question at issue": "What are the possible processes which can be carried out [by a computor] in computing a number?"

Turing imagines a computor writing symbols on paper that is divided into squares "like a child's arithmetic book". As the two-dimensional character of this computing space is taken not to be essential, Turing takes a one-dimensional tape divided into squares as the basic computing space and formulates one important restriction. That restriction is motivated by limits of our sensory apparatus to distinguish *at one glance* between symbolic configurations of sufficient complexity. It states that only finitely many distinct symbols can be written on a square. Turing suggests as a reason that "If we were to allow an infinity of symbols, then there would be symbols differing to an arbitrarily small extent", and we would not be able to distinguish at one glance between them. A second (and clearly related) way of arguing the point uses a finite number of symbols and strings of such symbols. E.g., Arabic numerals like 9979 or 9989 are seen by us at one glance to be different; however, it is not possible for us to determine immediately that 88899954967899998769 is different from 88899954967899998769.

Now we turn to the question: "What determines the steps of the computor, and what kind of elementary operations can he carry out?" The behavior is *uniquely* determined at any moment by two factors: (i) the symbols or symbolic configuration he observes, and (ii) his "internal state". This uniqueness requirement may be called the **determinacy condition** (D); it guarantees that computations are deterministic. Internal states or, as Turing also says, "states of mind" are introduced to have the computor's behavior depend possibly on earlier observations and, thus, to reflect his experience. Since Turing wanted to isolate operations of the computor that are "so elementary that it is not easy to imagine them further divided", it is crucial that symbolic configurations relevant for fixing the circumstances for the actions of a computor are immediately recognizable. So we are led to postulate that a computor has to satisfy two **finiteness conditions**:

(F.1) *there is a fixed finite number of symbolic configurations a computor can immediately recognize;*

(F.2)[3] *there is a fixed finite number of internal states that need be taken into account.*

For a given computor there are consequently finitely many different combinations of symbolic configurations and internal states. Since his behavior is according to (D) uniquely determined by such combinations and associated operations, the computor can carry out at most finitely many different operations. These operations are restricted as follows:

(O.1) *only elements of observed symbolic configurations can be changed;*

(O.2) *the distribution of observed squares can be changed, but each of the new observed squares must be within a bounded distance L of an immediately previously observed square.*

Turing emphasized that "the new observed squares must be immediately recognisable by the computer"; that means the distributions of the new observed squares arising from changes according to (O.2) must be among the finitely many ones of (F.1). Clearly, the

---

[3] Gödel objected in *1972* to this finiteness condition for a notion of human calculability that might properly extend mechanical calculability; for a computor or for a paper machine it is quite unobjectionable.

same must hold for the symbolic configurations resulting from changes according to **(O.1)**. Since some steps may involve a change of internal state, Turing concluded that the most general single operation is a change of *either* symbolic configuration and, possibly, internal state *or* observed square and, possibly, internal state. With this restrictive analysis of the steps of a computor, the proposition that his computations can be carried out by a Turing machine is established rather easily.[4] Thus we have **Turing's Theorem**: Any number theoretic function F that can be computed by a computor satisfying the determinacy condition **(D)** and the conditions **(F)** and **(O)** can be computed by a Turing machine.

**1.3 TURING'S THESIS.** Turing's analysis and his theorem can be generalized: **(D)** is not needed to guarantee the Turing computability of F in the theorem. Computors that do not satisfy **(D)** can be mimicked by non-deterministic Turing machines and thus, exploiting the reducibility of non-deterministic to deterministic machines, by deterministic Turing machines. That allows us to connect Turing's considerations with those of Church discussed in § 1.0: Consider an effectively calculable function F and a computor who calculates the value of F in a logic **L**; by Turing's generalized theorem F is Turing computable and thus recursive. This argument for F's recursiveness does no longer appeal to Church's Central Thesis; rather, such an appeal is replaced by the assumption that the calculation in the logic is done by a computor satisfying the conditions **(F)** and **(O)**. If that assumption is to be discharged, then a substantive thesis is needed again. And it is this thesis we call **Turing's Central Thesis**: a mechanical computor must satisfy the finiteness conditions **(F)**, and the elementary operations he can carry out must be restricted as conditions **(O)** require.

Church wrote in a 1937 review of Turing's paper when comparing Turing computability, recursiveness, and $\lambda$-definability: "Of these, the first has the advantage of making the identification with effectiveness in the ordinary (not explicitly defined) sense evident immediately ..." Turing's work provided for Gödel "a precise and unquestionably adequate definition of the general concept of formal system". In the historical and systematic context Turing found him-

---

[4] Turing constructed machines that mimic the work of computors directly and observed: "The machines just described do not differ very essentially from computing machines as defined in § 2, 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 [in our terminology: comput*or*]." But compare our discussion in § 3.1.

self, he asked exactly the right question: "What are the possible processes a human computor can carry out in computing a number?" The general problematic *required* an analysis of the idealized capabilities of a mechanical computor. Let us emphasize that the separation of conceptual analysis (leading to the axiomatic conditions) and rigorous proof (establishing Turing's Theorem) is essential for clarifying on what the correctness of his general thesis rests; namely, on recognizing that the axiomatic conditions are true for computors who proceed mechanically. We have to remember that clearly when engaging in methodological discussions concerning artificial intelligence and cognitive science. Even Gödel got it wrong, when he claimed that Turing's argument in the 1936 paper was intended to show that "mental processes cannot go beyond mechanical procedures".

**2 STEPS (towards the basic theory).** We focus on results that are central for recursion theory and computer science; these results reinforce our conceptual analysis and are frequently appealed to in support of Church's Thesis. If we take for granted a representation of natural numbers in the two-letter alphabet of Turing machines and a straightforward definition of when to call a number-theoretic function *(Turing) computable,* we can recast our earlier question: Why does this notion provide "an unquestionably adequate definition of the general concept of formal system"? Is it at all plausible that every effectively calculable function is Turing computable?

**2.0 NORMAL FORM.** It seems that a naive inspection of the very restricted notion of Turing computability should lead to "No!" as a tentative answer to the second and, thus, also to the first question. However, a systematic development of Turing computability convinces one quickly that it is a very powerful notion. One goes almost immediately beyond the examination of particular functions and the writing of programs for machines computing them; instead, one considers machines that correspond to operations on functions and that yield, when applied to computable functions, ones that are also computable. Three such functional operations are crucial, namely, composition, primitive recursion, and minimization. The latter operation allows an unbounded search for a solution to equations of the form $g(y,xi, \ldots ,x_n)=0$: given a computable function g such that for every $x\text{-}|, \ldots ,x_n$ there is a y with $g(y,xi, \ldots ,x_n)=0$ one can define (by minimization) a new computable function f(xi, ... ,xn) whose value for the indicated arguments is the smallest y with $g(y,xi, \ldots , x_n)=0$. This is usually indicated by $f(x\text{-}|, \ldots ,x_n)=\hat{}iy.g(y,xi, \ldots ,x_n)=0$.

Kleene established in *1936* the equivalence between Gödel's recursiveness (defined through an equational calculus) and "μ-recursiveness". The latter notion is characterized by an inductive definition that is obtained from that for the primitive recursive functions by adding a single clause for minimization as described above. This characterization of the recursive functions together with the closure of computable functions under the above functional operations and the computability of a few simple initial functions implies that *all recursive functions are computable.*

Conversely, the idea underlying Church's argument for the recursiveness of functions "calculable in a logic" (and also Kleene's argument for the μ-recursiveness of functions "calculable in Gödel's equational calculus") can be used to show that every total computable function is actually recursive! Gandy called Church's argument the "step-by-step argument": if the steps in "logical calculations" are recursive, then the functions being calculated are recursive. But in what sense can steps be "recursive", as they are taken in a logical calculus, whereas recursiveness is a property of number theoretic functions and predicates? It was Gödel who had shown in his 1931 paper, how to "code", "arithmetize", or - as we often put it in recognition of the fundamental character of his technique - "Gödel-number" the finite syntactic objects of a logical calculus. Given the arithmetization of syntactic objects, it is tedious, but not difficult to see that the syntactic notions (like formula or derivation) for "standard" formal theories are indeed (primitive) recursive. Computations of Turing machines can be described in exactly the same way to yield ultimately that *all Turing computable function are recursive.*

This result was generalized by Kleene, who associated partial recursive functions with computations; i.e., the domain of a function is now taken to coincide with the set of arguments for which the corresponding Turing machine computation terminates. The essence of the earlier observations is captured for partial recursive functions by Kleene's Normal Form Theorem. In the formulation (for a one-place function f to keep matters simple, but without loss of generality) one uses a particular three-place predicate T and a one-place function U. The T-predicate applied to numbers e, x, y expresses that y is the code of a computation of a Turing machine with code e for the numerical argument x; U extracts from the computation, in case it terminates, the numerical result. **Kleene's**

**Theorem** can now be stated as follows: For every partial recursive function f there is a natural number e, such that for all x in the domain of f, f(x) = U(|iy.T(e,x,y)). (The domains of f and the function A,x.|iy.T(e,x,y) are co-extensional.)

**2.1 UNCOMPUTABLE FUNCTIONS.** Kleenex Theorem exploits the uniform description of Turing machine computations and has most important consequences. Consider, first of all, the two-place function g(e,x)=U(|iy.T(e,x,y)); g is partial recursive and provides an enumeration of all one-place partial recursive functions (Kleene's Enumeration Theorem). Together with the fact that the partial recursive functions are exactly the Turing computable ones this theorem guarantees the existence of a universal Turing machine. Such a machine was explicitly constructed by Turing, and the idea underlying his construction was fundamental for the development of the architecture of digital computers through von Neumann in *1958.* How is the existence established by using Kleene's mathematical work? Note that by the above observations the two-place function g can be computed by a Turing machine M[g]. M[g] interprets its first argument as the code e of a Turing machine M[f] computing the one-place function f; its second argument is taken by M[g] as the argument for f or rather M[f]. Then M[g] proceeds to compute f(x) by following the program for M[f]; thus, M[g] is able to duplicate the computation of ANY Turing machine. - The existence of a universal machine is established here without the elaborate construction of Turing's; clearly, if one desires to do so, one can extract from the above argument a program for a universal machine.

For our purposes some "negative" results are most important, as they were for the pioneers of the subject. In contrast to Kleene's Enumeration Theorem for partial recursive functions, we can show by a classical diagonal argument[5] that the one-place total recursive functions cannot be enumerated by a *total* two-place recursive function. The argument proceeds as follows: Assume, to obtain a contradiction, that there is a total recursive function h enumerating all one-place total recursive functions f; i.e., for every function f there is a natural number e such that h(e,x)=f(x) for all x. Clearly, as h is (assumed to be) recursive the one-place function f* defined by

(1) $$f^*(x) = h(x,x)+1$$

[5] **The diagonal method of proof goes back to Cantor who used it first to show that the set of real numbers is not enumerable.**

is also recursive; thus, as h is an enumeration, we have for f* an e* such that for all x :

(2) $h(e^*,x)=f^*(x)$.

For x=e* we obtain from (2)

(3) $h(e^*,e^*)=f^*(e^*)$;

and from (1)

(4) $f^*(e^*)-h(e\backslash e^*)+1$.

Obviously, (3) and (4) imply the contradiction $h(e^*,e^*)=h(e^*,e^*)+1$. Thus we know that an enumeration function for the total recursive functions cannot be recursive. And similarly, functions enumerating Turing machines that compute total number theoretic functions cannot be calculated by Turing machines.

**2.2 UNDECIDABLE PROBLEMS.** A modification of the above argument (exploiting the existence of a universal machine) shows that particular questions concerning Turing machines cannot be answered by Turing machines. The most famous question is this: Does the computation of machine M for input x terminate or halt? This is the Halting Problem as formulated by Turing in *1936]* it is clearly a fundamental issue concerning computations. Turing used the unsolvability of this particular problem to establish the unsolvability of related machine problems, e.g. the Self-halting Problem and the Printing Problem. For that purpose Turing made implicit use of a notion of (effective) reducibility; a problem P, identified with a set of natural numbers, is reducible to another problem Q iff there is a recursive function f, such that for all x: P(x) iff Q(f(x)). Thus, if we want to see that x is in P we compute f(x) and test whether that number is in Q. In order to obtain his negative answer to the decision problem Turing reduced (in a most elegant way) the Halting Problem to the Decision Problem; thus, if the latter problem were solvable, the former problem would be.

The self-halting problem K is the simplest in an infinite sequence of increasingly complex and clearly undecidable problems, the so-called *jumps.* First of all notice that for a machine M with code e the set K can be defined arithmetically by the statement "there exists a y, such that $T(e,e,y)^{If}$. K is indeed *complete* for sets A that are definable by formulas obtained from recursive ones by just prefixing one existential quantifier; i.e. any such A is reducible to K. These A can be given a different and very intuitive characterization: A is either the empty set or the range of a recursive function. Under this characterization the A's are naturally called "recursively enu-

merable", or simply r.e.. It is not difficult to show that the recursive sets are exactly those that are r.e. and have an r.e. complement. Post gave a different way of "generating" these sets by production systems and thus opened a distinctive approach to recursion theory; such an approach is very beautifully presented in Smullyan's book *1961.* Now, coming back to more complex sets, to obtain the *jump hierarchy* the concept of computation is relativized to sets of natural numbers whose membership relations are revealed by "oracles". The jump $K^1$ of K, for example, is defined as the self-halting problem, when an oracle for K is available. This hierarchy can be associated to definability questions in the language of arithmetic: all jumps can be defined by increasingly complex arithmetical formulas, and all arithmetically definable sets are reducible to some jump.

**2.3 INCOMPLETE THEORIES.** The above considerations underly the "arithmetic hierarchy" introduced by Kleene and Mostowski. But there are sets of natural numbers that are not definable by arithmetic formulas; a particular example is the set of Gödel-numbers of sentences of true arithmetic statements. If that set were arithmetically definable, one could formulate arithmetically the "liar sentence" that expresses its own falsity. This observation of Gödel and Tarski is the cornerstone for proving that any semantically sound formal theory of arithmetic (with symbols for addition and multiplication) is incomplete: No matter which true statements we choose as axioms and no matter which inferences leading from true statements to true ones we select, there will be true statements that are not provable in the theory. That is correct as long as formality requirements are imposed on the theory, meaning - in mathematical terms and using the identification of formality with recursiveness - that the set of theorems is r.e.. (The theorem predicate of such a theory is the first example of a predicate that is r.e., but not recursive.)

This argument, under the explicit assumptions on the theory, establishes the existence of a true, but formally undecidable, sentence and reveals the purely recursion-theoretic character of the first incompleteness theorem. Gödel, in his 1931 paper, constructed a particular unprovable statement for the particular theory PM inspired by Russell and Whitehead's "Principia Mathematica", namely, the self-referential statement G expressing that it itself is not provable in PM. Recall that the latter theory was taken not only as a fundamental, but indeed universal theory for all of mathematics. Using an improvement of GödePs construction due to Rosser, one can

formulate a sentence R whose independence from PM is established under the sole assumption of PM's syntactic consistency. In this form the First Incompleteness Theorem is taken to refute an underlying assumption of Hilbert's Program, namely that formal theories like PM can capture "completely" mathematical practice. The aim of establishing the consistency of formal theories by restricted mathematical, so-called finitist, means was taken to be unreachable on account of Gödel's Second Incompleteness Theorem. This theorem states that the proposition "PM is consistent", when formulated in the language of PM, cannot be proved in PM. To reach a definite verdict on the original version of Hilbert's Program, it has only to be assumed that finitist mathematics can be formalized in PM.[6]

Given the background for Turing's work on the decision problem and Gödel's work on incompleteness, it is quite clear that they needed as broad a notion as possible. Turing's computability notion is highly idealized, because it disregards limitations on two resources, namely, space and time. The former is disregarded, as the number of tape squares scanned during a computation is unbounded; the latter is disregarded, as the number of computation steps is not limited. If we insist that the number of steps be bounded, we automatically insist on a bound for the number of tape squares that can be used in a computation. Yet it seems only too necessary to impose such bounds: Our lifespan is limited, and the size of the physical universe is bounded. It is an open problem, whether a mathematical concept can capture "feasible computations". For an informed judgement, conceptual analysis has to go hand in hand with experience derived from mathematical development and computational practice. This is a central issue of contemporary complexity theory.

**INTERLUDE: from mathematics to physics**. Gödel's First Incompleteness Theorem shows that most mathematically interesting theories contain undecidable statements. In the prototypical case of Peano arithmetic the existence of undecidable sentences was regarded for many years as irrelevant to the working mathematician. In 1977 Paris and Harrington exhibited an undecidable sentence in Peano arithmetic that has direct significance, at least, for the

---

[6] It is generally (and most plausibly) assumed that finitist mathematics can be formalized in elementary number theory and, possibly, coincides with a weak fragment of number theory, primitive recursive arithmetic.

mathematician working in Ramsey theory. Their sentence can be promptly decided in Zermelo-Fraenkel set theory with the axiom of choice (ZFC), i.e. the first-order theory most frequently adopted as the official foundation for the whole edifice of mathematics. However, ZFC is no less incomplete than Peano arithmetic: For instance, as shown by Gödel and Cohen, one cannot settle in ZFC Cantor's problem, whether there is a set whose cardinality lies strictly between the cardinality of the set of natural numbers and that of the set of real numbers.[7]

Even the founders of set theory were not unanimous on the problem, whether the evolution of their subject would follow that of geometry after the discovery of the independence of the parallel postulate. Consider the concluding remarks of Skolem's 1922 paper *Einige Bemerkungen zur axiomatischen Begründung der Mengenlehre*:

> The most important result above is that set theoretic notions are relative. I had already communicated it orally to F. Bernstein in Göttingen in the winter of 1915-16. There are two reasons why I have not published anything about it until now: first, I have in the meantime been occupied with other problems; second, I believed that it was so clear that axiomatization in terms of sets was not a satisfactory ultimate foundation of mathematics that mathematicians would, for the most part, not be very much concerned with it. But in recent times I have seen to my surprise that so many mathematicians think that these axioms of set theory provide the ideal foundation for mathematics; therefore it seemed to me that the time had come to publish a critique.

This remark was preceded by the question: "What does it mean for a set to exist if it can perhaps never be defined?" Skolem gave a most interesting answer to it. "It seems clear that this existence can be only a manner of speaking, which can lead only to purely formal propositions - perhaps made up of very beautiful words - about objects called sets. But most mathematicians want mathematics to deal, ultimately, with performable computing operations ..." There are related developments in mathematics and proof theory traceable to Skolem and contemporaneous work of Weyl in *Das Kontinuum*, exploring the formalization of the scientifically applicable parts of set theoretic mathematics in weak formal theories. And these theories are very much motivated by computational concerns; cf. [Feferman].

It is possible that from a long process of natural selection Euclidean-like set theoretic vertebrates, or even primates, will

---

[7] For informative expositions see the relevant chapters in the *Handbook of Mathematical Logic* edited by Barwise.

emerge. But, perhaps, the first-order treatment of sets parallels the Hamiltonian treatment of the forced pendulum, where the official a priori determinism is made ineffective by a teeming microcosm of capricious details. If that were so, the incompleteness of ZFC - in the framework of a complete logic - may be ascribed to uncontrollable, uninteresting details in our set theoretic stipulations/observations, just as unpredictability of a forced pendulum - in the framework of deterministic Hamiltonian mechanics - is caused by uncontrollable, uninteresting microscopic perturbations in the preparation/measurement of the system. A forced pendulum is a small sphere attached to the bottom end of a string; the sphere may oscillate in any direction, while the top end of the string is forced to oscillate along a horizontal line under the action of a crank shaft. Assume that the forcing frequency is a little higher than the natural frequency of the pendulum. Initially, the sphere will oscillate in parallel to the forcing oscillations, but then a perpendicular component of motion appears. Eventually, the motion becomes stationary along a circle with the period of the rotation being equal to the frequency of the forcing oscillation. But however precise we try to make our knowledge of the initial conditions of the forced pendulum, and notwithstanding the deterministic character of the Hamiltonian theory describing the forced pendulum, we cannot predict, whether it will eventually rotate clockwise or counterclockwise.

Unpredictability follows from the fact that a multitude of infinitesimal perturbations in the initial conditions causes macroscopic bifurcations already in the short-term evolution of the system. Bifurcation entails a sort of incompleteness of the underlying deterministic theory, at least concerning the problem of inferring from the initial conditions a precise truth value for the proposition "The system shall eventually rotate clockwise". The best technology can't help the Hamiltonian mountain to bring forth the appropriate clockwise or counterclockwise mouse. The reader will recall that Plato and Galileo had different views on the readability of the Book of Nature: The former believed that the plethora of accidental perturbations affecting the physical world and our perception of it would not allow any experimental physical theory at all. The latter regarded the *Book of Nature* as written in terms of triangles, circles, and other geometric figures so that we can read it "provando e riprovando" (by repeated experiment).

A typical reaction to incompleteness phenomena in mechanics is to regard them as unphysical - just as undecidable sentences in

mathematics are sometimes regarded as not genuinely mathematical. This attitude is made explicit in quantum statistical mechanics by the slogan "Nature does not have ideals". In other words, whenever a physical system is described by an algebra A of operators, A should have no nontrivial quotient structure. One of the effects of [Mundici 1986] is that, for the algebras of operators describing the thermodynamic limit process in quantum statistical mechanics, one has a reasonable notion of axiomatic presentation of the system. It may be the case that the axiomatization is essentially Gödel incomplete - in the sense that, although there is a recursive listing of the axioms, there is no decidable extension. In this case the system must have a very stubborn quotient structure, one that can only be eliminated by destroying the effective presentability of the system. As for Peano's axiomatization of arithmetic essential Gödel incompleteness entails the incompatibility of two desiderata, namely:

(a) completeness of the information available within the formalization, and

(b) effective computability of (the consequences of) this information.

In this context it may be of interest to mention that the undecidability of the halting problem has been used by da Costa and Doria to show the undecidablity of problems in classical mechanics and the incompleteness of suitable axiomatizations of the theory; see [Da Costa-Doria, 1993] and references therein.

**3 PHYSICAL LIMITS.** We discussed limits of computations in the logical sense first and were concerned with the Incompleteness Theorems and the unsolvability of the Entscheidungsproblem. Now we want to give "flesh" to the abstract machines and ask: What are general physical constraints on computational devices? Recall that Turing appealed in his analysis to the limitations of the sensory apparatus of human computors; however, he claimed that the justification for his (central) thesis lies ultimately "in the fact that the human memory is necessarily limited". This remark is not expanded upon at all, and we can only speculate as to Turing's understanding of this "fact": Did he have in mind more than the spatial limitations for "encoding" finite configurations (discussed below)? If such limitations hold also for computing devices, are there ways of getting around their effect, e.g. by speeding up operations or by using more complex ones? We begin our considerations with a computational problem concerning finite graphs.

**3.0 ONE PHYSICAL STEP.** Draw an edge between each pair of distinct vertices of a hexagon. The resulting graph is known as the 6-clique and is denoted by K6. Now follow freely your inspiration and color the 15 edges of K6 red or blue. Then, necessarily, there will exist a monochromatic 3-clique, i.e., a triangle with only blue or only red sides. For a proof of this claim choose your favorite vertex V in K6 and suppose for the moment that there are at least three red edges VA, VB, VC. If the triangle ABC is blue, we are done, otherwise one of the triangles VAB, VAC, or VBC must be red, and we are done. What, if our hypothesis fails? In this case V will be a vertex with at least three blue edges, and a photocopy of the above argument, possibly printed in red ink, yields a monochromatic triangle. Q.E.D.

The argument provides immediately a direct and fast way of finding a monochromatic triangle for each possible coloring of the hexagon. Passing from six to fortyfive points and coloring all edges of the 45-clique K45 red or blue, consider the question, whether all such colorings have a monochromatic 5-clique. Experimentally, the answer seems to be affirmative, yet nobody has an argument to exclude the existence of a counterexample - and nobody is willing to check all possible colorings.[8] Here is the description of a "concrete" Gedanken-Turing-machine T that decides in less than 60 minutes, whether there is an exceptional coloring of K45, i.e., a red-blue coloring having no monochromatic 5-cliques:

(1) T systematically tries all possible $n=2^{990}$ colorings of K45 until an exceptional coloring is found, in which case T rings a bell and stops;
(2) T's instructions are written in such a way that for each individual coloring T decides in less than half an hour, whether the coloring has a monochromatic 5-clique: the number of 5-cliques to check is just $45!/(40!x5!)=1221759$;
(3) Then we accelerate the tape of T, as is frequently done in comic movies, so that the second coloring is checked in (1/4)-th, the third in (1/8)-th,..., the n-th in $1/(2^n)$-th of an hour;
(4) In this way T solves the problem in less than 60 minutes, as required.

---

[8] This problem originated from a theorem of Ramsey and is one among many famous problems that can be easily explained, but that cannot be solved even by the experts of the Government with their latest supercomputers.

What is wrong with this T? Does relativity theory impose an insurmountable upper bound on the number of steps T can perform in one second? Does quantum thermodynamics impose a lower bound on the amount of heat being produced by T during the computation? We can argue as follows: Suppose T comes down from our mathematical world, where it can at most assume a papery nature, to the physical world satisfying the following two conditions:
(i) time is not recycled, i.e., no portion of the time used for one step can be used for another step - this is a reformulation of the sequential behavior of Turing machines;
(ii) energy is not recycled, i.e., no portion of the energy used for one step can be used for another step.
Then, if f is the number of steps performed by T in one second (T's frequency), and if W is the power (energy per second) used by T measured in watt, T will obey the inequality

$$f^2 \leq (2\pi W)/h,$$

where $h=6.6256\text{x}10^{-34}$ joule x sec is Planck's constant; that means the power absorbed by T grows at least as fast as the square of its frequency. The argument for the inequality is roughly this[9]: The portion of the tape that is scanned during a computation step undergoes a noticeable modification of its physical properties. Hence, by the Heisenberg inequality, the energy uncertainty AE of the tape square must be greater than h/(2rcAt), where At=1/f is the time needed for the step. A fortiori, the energy used for the step must be greater than AE, and the inequality immediately follows from (i) and (ii).

Albeit small, the multiplicative constant h has an effect, and the quadratic lower bound for W can be used as a convincing argument for the unfeasibility of (3) for Turing machines. Of course, one might argue that parallel computers are able to circumvent condition (i), and that a carefully designed Turing machine could, at least partially, circumvent condition (ii). More radically, one might argue that Heisenberg's uncertainty principle need not imply that any amount of energy is "used" for a computation step. In any case, no real computer has so far violated the above lower bound. In the next subsection we want to explore, how space-time features of computations are physically constrained, and how such constraints prevent us from having "arbitrarily" complex operations.

---

9 For details cf. [Mundici 1981].

**3.1 ONE PHYSICAL REGION.** In the above analysis of steps, there is no allusion to the details of the (physical) construction of Turing machines - except that each step requires an interaction between the square being scanned and the scanning head. Recall from section 1.2 that the restricted formulation of Turing machines achieves a mathematically uniform and simple description of computations and recall further that its adequacy is guaranteed by Turing's Theorem. The starting point of the analysis was, however, the "mechanical behavior" of a human computor operating on finite configurations. This behavior can be described directly and mathematically precisely.

But first we take a preliminary step and replace the states of mind by, what Turing described as, a "physical and definite counterpart". This is done by considering "states of mind" not as a property of the working computor, but rather as part of the configuration on which he operates. Turing discussed this replacement very vividly in section 9, III, of his 1936 paper:

**It is always possible for the computer [i.e., in our terminology, the computor] to break off from his work, to go away and forget all about it, and later to come back and go on with it. If he does this he must leave a note of instructions (written in some standard form) explaining how the work is to be continued. This note is the counterpart of the "state of mind". We will suppose that the computer works in such a desultory manner that he never does more than one step at a sitting. The note of instructions must enable him to carry out one step and write the next note.**

Mathematically that is done quite beautifully in [Davis 1958]: instantaneous descriptions of machines, that means finite sequences in the alphabet of a Turing machine, contain exactly one state symbol indicating by its position in the sequence which symbol is being scanned; programs of Turing machines can then be viewed as a set of Post production rules operating on (a single symbol of) such instantaneous descriptions.

If in this way of describing Turing machines one replaces finite sequences by finite graphs (with a few well-motivated properties) and the simple Post-Turing operations on one symbol at a time by operations on a fixed finite number of "distinguished" graphs, then one arrives at the notion of a *Kolmogorov Machine.* This latter notion, or rather a general concept of algorithm, was introduced by Kolmogorov and Uspensky in 1958; for an informative discussion see [Uspensky 1992]. As it turns out KMs compute exactly the Turing computable number theoretic functions, i.e. the partial recursive functions. The focus on a fixed finite number of distinguished graphs can be motivated by thinking of a human computor operating

according to rules; and with this understanding, as is detailed in [Sieg and Byrnes], the KMs provide what we called above a mathematically direct and precise description of Turing's computor. But the restriction can also be motivated by physical considerations; let us look at the underlying relativistic limitations.[10]

Assume that a machine operates on configurations containing z different "symbols", each symbol being physically represented or coded by at least one atom; that is an altogether reasonable assumption. Then there must be at least z pairwise disjoint regions containing the codes (of the symbols). Otherwise, the electron clouds of two different codes might overlap, making the codes indistinguishable and leading the machine to "mental confusion". Let c and a denote, respectively, the speed of light and Bohr's radius of the hydrogen atom, where $a/c=0.176x1\,CM^{8}$ seconds. It follows that the codes will be contained in a volume V of at least $z(4/3)7ca^3$ cubic meters; that forces the diameter 2r to be larger than $2az^{1/3}$ meters - the diameter being the largest possible distance between two codes in this volume. Let f be again the frequency of our machine and note that 1/f is the time available for each computation step. Since signals cannot travel faster than light and since a computation step involves the whole configuration, it follows that f cannot exceed $c/(2az^{1/3})$ steps per second. Thus, we obtain the inequality $f_z i/3 < 2.828x10^{18}$ steps per second, which points out a fundamental incompatibility between high number of codes (thus, size of configurations) and high computational speed. The operations of KMs are thus necessarily restricted in complexity, as they have to lead from distinguished graphs to distinguished graphs; and within the given physical boundaries only finitely many different graphs are realizable.

**3.2 CELLS & CIRCUITS.** One alternative to speeding up computations was mentioned already: parallel operations. *Cellular automata* introduced by Ulam and von Neumann operate in parallel; a particular cellular automaton was made popular by Conway, the so-called game of life. A cellular automaton is made up of many identical cells. Typically, each cell is located on a regular grid in the plane and carries one of two possible values, say, 0 or 1; after each time unit its values are updated according to a simple rule that

[10] We follow Mundici [1981], pp.302 ft, Mundici [1983], pp. 43 ft, and Sieg [1994], section 3.3, for the connection to Turing's claim that memory limitations provide the justification for the finiteness conditions.

depends only on its own previous value and the previous values of its neighboring cells. Such cellular automata can simulate universal Turing machines; they also yield simulations of very general and complex physical processes. It should be noted that cellular automata do not satisfy the finiteness axioms for Turing's computor. The reason is that computation steps may operate on unbounded regions of the plane. But that does not mean that cellular automata cannot be simulated by Turing machines: Indeed, they can be!

von Neumann used these devices to construct (complicated) self-replicating machines. But it is possible to avoid a detailed construction of von Neumann's to create a "self-reproducing" machine, i.e. Turing machines. First one notices the possibility of performing operations on machines effectively (indeed, primitive recursively) on the codes of their programs; the most pervasive operation is captured in Kleene's S-m-n-Theorem that allows to determine the code of a machine for n arguments from a machine for (m+n) arguments, when m arguments are fixed. Then one can establish a fundamental fact, the so-called Recursion or Fixed-Point Theorem. And that allows the proof of the existence of a self-replicating automaton in the following sense: there is an m, such that g(m,y)=m for all y. For details concerning these arguments we have to point to the literature; a very good presentation can be found in [Davis 1958] or [Cutland].[11]

Another interesting model of parallel computation is provided by Boolean circuits. The very practical justification for this model is that the building blocks of most real machines are Boolean circuits. In their simplest form Boolean circuits are given by formulas of the propositional calculus. Any such formula F of q variables outputs 1 or 0 according to whether the assignment $b_1,\ldots,b_q$ satisfies F or not. The most general formulas contain apart from "and", "or", "not" also other connectives, such as "iff". Furthermore, while the output value of a subformula other than a variable may serve as the input for exactly one (larger) subformula, the truth value of a subcircuit is addressed to many other subcircuits simultaneously. The only requirement is that there are no loops. In this way the input information $b_1,\ldots,b_q$ is processed in parallel by many subcircuits, each sending their outputs to other subcircuits, and finally we obtain the output. In contrast to Turing machines, which can handle

---

[11] In [Odifreddi, pp. 170-174] one finds a most informative and wide-ranging discussion on self-reproduction and cellular automata.

inputs of unbounded length, a Boolean circuit accepts as its input only sequences of q bits. There are $2^q$ such sequences and $2^{2^q}$ 0-1-valued functions of q variables; most of them are irreducible, in the sense that the best Boolean circuits computing the function are not very different from the (trivial) listing of the values of the function over each possible input. (This fact was first proved by Shannon.) On the other hand, there do exist seemingly complex 0-1-valued Boolean functions that can be represented by short circuits. It is a very interesting problem to write down the shortest possible circuits; curiously enough, nobody knows the shortest circuits for addition and multiplication. In fact, very little is known about the computing power of Boolean circuits.

**3.3 GENERAL PARALLELISM.** How can parallelism be captured in a general mathematical way, not restricted to the simple pattern of cellular automata or Boolean circuits, but clearly encompassing them? Gandy provided for the first time in his *1980* a conceptual analysis and a general description of parallel algorithms. These algorithms are thought to be carried out by "discrete deterministic mechanical devices", i.e., machines satisfying the physical assumptions explicit in our discussion in the last section. As to such "mechanical devices" Gandy suggested that "the reader may like to imagine some glorious contraption of gleaming brass and polished mahogany, or he may choose to inspect the parts of Babbage's 'Analytical Engine' which are preserved in the Science Museum at South Kensington". And, to give the above "i.e.-remark" in Gandy's language, "The only physical assumptions made about mechanical devices ... are that there is a lower bound on the linear dimensions of every atomic part of the device and that there is an upper bound (the velocity of light) on the speed of propagation of changes". He formulated axiomatic principles for these devices and proved that whatever can be calculated by devices satisfying the principles is also computable by a Turing machine; this is a marvel of analysis, though not of exposition. The definitional preliminaries are lengthy; Shepherdson wrote [in Herken 1988]: "Although Gandy's principles were obtained by a very natural analysis of Turing's argument they turned out to be rather complicated, involving many subsidiary definitions in their statement. In following Gandy's argument, however, one is led to the conclusion that that is in the nature of the situation ..."

We want to give an informal description that skirts the preliminary definitions, focusing rather on the intuitive considera-

tions that underly the mathematical formulations. The configurations on which a Gandy machine GM operates are taken to be hereditarily finite sets over some (potentially infinite) set A of urelements as labels, HF(A). The configurations must satisfy two boundedness conditions: the first, called *Limitation of Hierarchy*, expresses that all configurations of a given GM must be in a segment of the cumulative hierarchy HF(A) containing only sets of rank less than a fixed natural number; the second, called *Unique Reassembly*, requires that all configurations can be uniquely reassembled from parts of bounded size. The third and central condition, called the *Principle of Local Causation*, governs the transition from one configuration $C_n$ to the next one $C_{n+1}$: $C_n$ can be reassembled (according to condition 2) from parts of bounded size that fit into a fixed finite number of isomorphism types; GM operates on these parts in parallel and assembles $C_{n+1}$ from locally computed parts. It is here that the relativistic limitation on the speed of light comes in or, more generally, as Gandy puts it: "Its justification (i.e., that of the Principle of Local Causation) lies in the finite velocity of propagation of effects and signals: contemporary physics rejects the possibility of instantaneous action at a distance." This forces the restriction to parts of bounded size on which the computation is carried out.

The successive states of cellular automata (and the successive configurations of Gandy machines) can be computed by suitable Turing machines; but how complex is this "serialization"? This is a most interesting question, as the cellular automata are particular kinds of dynamical systems that are used to simulate physical processes! Indeed, Fredkin has been advocating the use of (reversible) cellular automata in physics for some time. In his *Digital Mechanics* he conjectures "that there will be found a single cellular automaton rule that models all of microscopic physics; and models it exactly." The interested reader should delve into the paper by Richard Feynman, *Simulating physics with computers*, published in the International Journal of Theoretical Physics, volume 21 (1982), pp. 467-488. See also [Herken 1988] and [Toffoli and Margoulis].

**EPILOGUE.** Every mathematical model of physical processes comes with at least two problems: How accurately does the model capture physical reality, and how efficiently can the model be used to make predictions? What is distinctive about the modern developments is this: Computer simulations have led to an emphasis of the algorithmic aspect of scientific laws and, conversely, physical systems are being considered as computational devices that process information

much as computers do. Let us compare the forced pendulum (or the classical pendulum) with a Gedanken-45-clique (with a 6-clique, respectively), whose red-blue coloring is assumed to vary with time. Just as a classical exercise in Hamiltonian mechanics immediately yields conservation of energy for the pendulum, similarly the combinatorial argument given at the beginning of section 3.0 yields the conservation of a monochromatic 3-clique in the red-blue 6-clique. By contrast, even the most powerful computer will not be able to decide whether a given forced pendulum will eventually rotate clockwise, or whether a variable red-blue 45-clique will always have a monochromatic 5-clique.

Computational intractability stems, perhaps, from the impossibility of conceptually handling (symbolizing) a virtually infinite amount of relevant information, namely, the list of digits of the real numbers measuring the initial conditions of the pendulum or the list of red and blue edges in all possible colorings of the 45-clique. Owing to this impossibility, the forced pendulum, as well as the variable red-blue 45-clique remain their own best simulators. The situation is similar for cellular automata: in some fortunate cases a mathematical theorem can completely describe the evolution of a cellular automaton, but in general an automaton is its own best simulator. When computing a Boolean function, what should be considered the "ratio extrema[11], namely the direct consultation of the table of all its values, almost always turns out to be the "ratio sola unica". When no shortcuts are available and the computer attains the highest degree of resemblance with the simulated system, we regard the system (or, equivalent^, the simulation) as being irreducible.

The Turing machine model is not well suited to simulations of irreducible systems, as most physical processes seem to correspond to parallel computations. Even non-deterministic Turing machines are unsuitable, because they require too much parallelism; e.g., in the search for an exceptional coloring of the 45-clique we would replace the above Turing machine T by $2^{990}$ Turing machines all working in parallel, each checking a different coloring. Setting up paradigms for parallel computing is expected to afford a more efficient handling, not only of simulations of physical processes, but also of combinatorial and decision problems. It seems, ironically, that our mathematical inquiry into *paper machines* has led us to a point where (effective) mathematical descriptions of nature and (natural) computations for mathematical problems coincide.

## BIBLIOGRAPHY


J. Barwise (ed.), *Handbook of Mathematical Logic*, North-Holland, Amsterdam, Fifth Reprinting, 1991.

A. Church, An unsovable problem of elementary number theory, American Journal of Mathematics 58, 1936, pp. 345-363. Also in (Davis 1965).

A. Church, Review of *Turing 1936*, Journal of Symbolic Logic 2, 1937, pp. 42 -43.

N.C.A. da Costa, F.A.Doria, On Arnold's Hilbert Symposium problems, In: Proceedings of the 3rd Gödel Colloquium in Brno, August 1993, G. Gottlob et al. (eds.), Springer Lecture Notes in Computer Science, 713, 1993, pp. 152-158.

N. Cutland, *Computability - An introduction to recursive function theory*; Cambridge University Press, 1980.

M. Davis (ed.), *The Undecidable*; Raven Press, Hewlett (New York), 1965.

M. Davis, *Computability and Unsolvability*; McGraw-Hill, New York, 1958.

S. Feferman, Why a little goes a long way: Logical foundations of scientifically applicable mathematics; to appear in: *PSA 1992*, volume II, Proc. of the Philosophy of Science Association meeting, Chicago October 29-November 1, 1992.

Edward Fredkin, Digital mechanics; Physica D 45 (1990), pp. 254-270.

R. O. Gandy, Church's thesis and principles for mechanisms; In: *The Kleene Symposium*, J. Barwise, H.J. Keisler, and K. Kunen (eds.), North-Holland, Amsterdam, 1980, pp. 123-148.

R. O. Gandy, The confluence of ideas in 1936, In: [Herken 1988], pp. 55-111.

M.R. Garey, D.S. Johnson, *Computers and Intractability*, W.H.Freeman, San Francisco, 1979.

K. Gödel, Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I, Monatshefte für Mathematik und Physik 38, 1931, pp. 173-198. Also in [Gödel 1986].

K. Gödel, The present situation in the foundations of mathematics, lecture presented in December 1933, to appear in: [Gödel 1994].

K. Gödel, Über die Länge von Beweisen, Ergebnisse eines mathematischen Kolloquiums 7, 1936, pp. 23-24. Also in: [Gödel 1990].

K. Gödel, Remarks before the Princeton bicentennial conference on problems in mathematics, 1946; reprinted in: [Gödel 1990], pp. 150-153 .

K. Gödel, Some remarks on the undecidability results; written in 1972, published in: [Gödel 1990], pp. 305-306

K. Gödel, *Collected Works*, volume I; Oxford University Press, 1986.

K. Gödel, *Collected Works,* volume II; Oxford University Press, 1990.

K. Gödel, *Collected Works,* volume III; Oxford University Press, 1994; to appear.

R. Herken (ed.)$_f$ *The Universal Turing Machine (A half-century survey),* Oxford University Press, 1988.

D. Hilbert, P. Bernays, *Grundlagen der Mathematik,* volume II, Springer Verlag, Berlin, 1939.

S.C. Kleene, General recursive functions of natural numbers, Mathematische Annalen, volume 112 (5), 1936, pp. 727-742; reprinted in [Davis 1965], pp. 237-253.

S.C. Kleene, *Introduction to metamathematics,* Groningen, 1954.

D. Mundici, Irreversibility, uncertainty, relativity and computer limitations, Il Nuovo Cimento, Europhysics Journal, 61 B, n.2 (1981) pp. 297-305.

D. Mundici, Natural limitations of decision procedures for arithmetic with bounded quantifiers, Archiv für mathematische Logik und Grundlagenforschung 23 (1983), 37-54.

D.Mundici, Interpretation of AF C*-algebras in Lukasiewicz sentential calculus, Journal of Functional Analysis, 65 (1986) pp. 15-63.

P. Odifreddi, *Classical Recursion Theory;* North-Holland Publishing Company, volume I, second reprinting 1992; volume II, 1993.

R. Peter, *Rekursive Funktionen,* Verlag der Ungarischen Akademie der Wissenschaften, Budapest, 1951.

E. Post, Finite combinatory processes. Formulation 1, J. Symbolic Logic 1 (1936), pp. 103-105. Also in: [Davis 1965].

W. Sieg, Mechanical procedures and mathematical experience; to appear in: *Mathematics and Mind,* A. George (ed.), Oxford University Press, 1994, pp. 71-117.

W. Sieg, J. Byrnes, Turing's argument: a mathematical explication; manuscript.

T. Skolem, Begründung der elementaren Arithmetik durch die rekurrierende Denkweise ohne Anwendung scheinbarer Veränderlichen mit unendlichem Ausdehnungsbereich; Videnskapsselskapets skrifter, I. Matematisk-naturvidenskabelig klasse, no. 6; translated and reprinted in [van Heijenoort 1967], pp. 302-333.

R. Smullyan, *Theory of Formal Systems,* Princeton University Press, 1961.

G. Tamburrini, *Reflections on Mechanism;* Ph.D. Thesis, Columbia University, 1988.

T. Toffoli, N. Margoulis, *Cellular Automata Machines,* MIT Press, Cambridge, Massachusetts, 1988.

A. Turing, On computable numbers, with an application to the Entscheidungsproblem, Proc. London Mathematical Society 42, 1936, 230-265. Also in: [Davis 1965].

**V.A. Uspensky, Kolmogorov and Mathematical Logic; Journal of Symbolic Logic, 37 (2) (1992), 385-412.**

**J. van Heijenoort (e<±), *From Frege to Gödel;* Harvard University Press, Cambridge, 1967.**

**J. von Neumann, Zur Hilbertschen Beweistheorie, Mathematische Zeitschrift 26, 1927, pp. 1-46.**

**J. von Neumann, *The Computer and the Brain;* Yale University Press, 1958.**