A Theory of Computation and Complexity over the Real Numbers

Lenore Blum

TR-90-058

October, 1990

ABSTRACT

The classical theory of computation and complexity presupposes all underlying spaces are countable and hence ipso facto cannot handle arbitrary sets of real or complex numbers. Thus e.g., Penrose (1990) acknowledges the difficulty of formulating classically his question "Is the Mandelbrot set recursive?" On the other hand, this as well as a number of other inherent questions of decidability and computability over the reals or complex number can be naturally posed and settled within the framework presented in this paper.

A Theory of Computation and Complexity over the Real Numbers

Lenore Blum[1]

International Computer Science Institute, 1947 Center Street, Berkeley, CA 94704; Mills College, Oakland, CA 94613; and University of California, Berkeley, CA 94720, USA (lblum@icsi.berkeley.edu)

1 Introduction

Classically, the theories of computation and computational complexity deal with discrete problems, for example over the integers, about graphs, etc.. On the other hand, most computational problems that arise in numerical analysis and scientific computation, in optimization theory and more recently in robotics and computational geometry, have as natural domains the reals **R**, or complex numbers **C**. A variety of ad hoc methods and models have been employed to analyze complexity issues in this realm, but unlike the classical case, a natural and invariant theory has not yet emerged. One would like to develop theoretical foundations for a theory of computational complexity for numerical analysis and scientific computation that might embody some of the naturalness and strengths of the classical theory.

Toward this goal, we have been developing a new theory of computation and complexity which attempts to integrate key ideas from the classical theory in a setting more amenable to problems defined over continuous domains. The approach taken here is both algebraic and concrete; the underlying space is an arbitrary commutative ring (or field) and the basic operations are polynomial (or rational) maps and tests. Thus, a real number is viewed as an entity in its own right, not as a decimal approximation, and the assumption made is that we can add and multiply (and divide) real numbers. By maintaining fundamental mathematical operations as primary, rather than reducing all computations to bit operations, the algebraic and dynamic structure of algorithms become apparent. For example in this model, as in practice, Newton's method for finding zero's of a polynomial f is performed on an arbitrary real number, not just a computable real, and the fundamental component of the algorithm is the rational operation $N_f(z) = z - \Big(f(z)/f'(z)\Big)$, not bit operations.

The classical theory of computation and complexity presupposes all underlying spaces are countable and hence ipso facto cannot handle arbitrary sets of real or complex numbers. Thus e.g., Penrose (1990) acknowledges the difficulty of formulating classically his question "Is the Mandelbrot set recursive?" On the other hand, this as well as a number of other inherent questions of decidability and computability over the reals or complex numbers can be naturally posed and settled within the new framework.

Our approach yields results in the continuous setting analogous to the pivotal classical results of *undecidability* and *NP-completeness* over the integers, yet reflecting the special mathematical

[1] This work was partially supported by National Science Foundation grants CCR-8712121 and CCR-8907663 and the Letts-Villard Chair at Mills College.

character of the underlying space. For example, over the reals we have that (1) the Mandelbrot set as well as most Julia sets are undecidable[2] and (2) the problem of deciding if an algebraic variety has a real point is *NP*-complete. While there are many subtle differences between the new and classical results, the ability to employ mathematical tools of more mainstream mathematics (such as from algebra, analysis, geometry and topology) in the domain of the reals may suggest new approaches for tackling the classical, as well as new, "*P=NP*?" questions.

The material covered here is based in large part on (Blum, Shub and Smale 1989) denoted in this paper by BSS, (Blum and Smale 1990) and (Blum 1990). Discussions of related work are contained in those papers. See also (Shub 1990a, 1990b) and (Smale 1990). Additional relevant literature is listed in the References.

2 Computable Functions and Decidable Sets

The classical theory of computation had its origins in work of logicians -of Gödel (1931), Turing (1937), Church (1936), Kleene (1936), Post (1936)- in the 1930's. Of course there were no computers at the time; this work, in particular Turing's, clearly anticipated the development of the modern digital computer. But even more, a primary motivation for the logicians was to formulate and understand the concept of *decidability*, or of a *decidable* set, thus to make sense of such questions: "Is the set of theorems of arithmetic decidable?" or "Is the set of polynomials with integer coefficients and integer solutions decidable?"[3]

Intuitively, a set $S \subset U$ is *decidable* if there is an "effective procedure" that given any element u of U (some natural universe) will decide in a finite number of steps whether or not u is in S, i.e. if the characteristic function of S (with respect to U) is "effectively computable." The models of computation designed by these logicians were intended to capture the essence of this concept of effective procedure/computation. The idea was to design formal "machines" with operations, and finitely described rules for proceeding step by step from one operation to the next, so simple and constructive that it would be self-evident that the resulting computations were effective.

In each formalism (e.g. Turing's), a function f from the natural numbers $\mathbf{N}$ to $\mathbf{N}$ is defined to be *computable* if it is the *input-output* function of some such machine (e.g. a Turing machine). It is quite remarkable that even though the formalisms were often markedly different, in each case, the resulting class of computable functions (and hence decidable sets) was exactly the same. Thus, the class of computable functions appears to be a natural class, independent of any specific model of computation.[4] This gives one a great deal of confidence in the theoretical foundations of the theory of computation. Indeed, what is known as *Church's thesis* is an assertion of belief that the classical formalisms completely capture our intuitive notion of computable function. Compelling motivation clearly would be required to justify yet a new paradigm.

---

[2] Indeed, the complements of these sets provide examples of *semi-decidable* sets that are undecidable over the reals.

[3] It was originally taken for granted (by mathematicians in general, and Hilbert (1901–1902) in particular) that the answers to these questions were both affirmative. The queries were actually posed as tasks: "Produce decision procedures for the given sets." The incompleteness/undecidabilty results of Gödel (1931) in the first place, and of Matijasevich (1971) on the unsolvabilty of Hilbert's Tenth Problem in the second, show such tasks cannot be carried out in full generality.

[4] These functions are often called the *(partial) recursive functions*.

3 Examples

In order to motivate our theory, we briefly discuss three examples, one from complex analytic dynamics, one from numerical analysis and one from classical complexity theory.

1 Is the Mandelbrot Set Decidable?

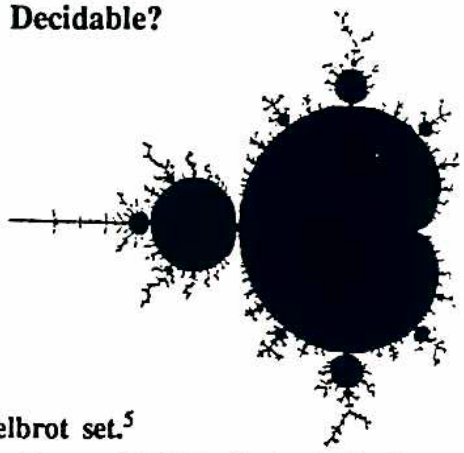

FIGURE 1 The Mandelbrot set.[5]

This question was asked by Penrose (1989) in his book *The Emperor's New Mind*. Recall the Mandelbrot set M can be defined:

$$M = \{c \in \mathbf{C} \mid p_c^n(0) \nrightarrow \infty\},$$
where $p_c(z) = z^2 + c$ and p_c^n is the *nth iterate of* p_c.

It is well known (se e.g. (Branner 1989)and (Douady and Hubbard 1984, 1985)) that the boundary of M has a rich and extraordinarily complex structure. Hence, the reasonableness of Penrose's query.

In a trivial sense the answer to the question is "yes" since classically, decidable sets, being subsets of **N** or some structure that can be "effectively coded" in **N**, must be countable. However, one might wish for a more satisfactory way to deal with such questions about subsets of $\mathbf{R}^n$ (for M, we are viewing **C** as $\mathbf{R}^2$). One way might be to consider the rational or algebraic skeletons of the sets in question. Problems quickly arise with this approach (e.g. consider the rational skeleton of the points on the curve $x^3 + y^3 = 1$ in the positive orthant). Another way might be to take a recursive analysis approach. For example, we might imagine a Turing machine being input a real number bit by bit by oracle. Using its internal instructions, the machine operates on what it sees, possibly every so often outputing a bit. The resulting sequence, if any, would be considered in the limit the (binary expansion of the) real output. Problems arise here when one wants to decide if two numbers are equal.

Penrose speculates on various such approaches and concludes (p.129) " One is left with the strong feeling that the correct viewpoint has not yet been arrived at."

---

[5] This illustration is from (Penrose 1989) and is reproduced with permission by the publisher.

2 The Newton Machine

Newton's method is perhaps the "algorithm" sine qua non of numerical analysis and scientific computation. Here we briefly recall Newton's method for finding zeros of polynomials in one variable.

Given a polynomial $f(z)$ over the complex numbers $\mathbf{C}$, define the *Newton map* $N_f : S \to S$ of the Riemann sphere $S = \mathbf{C} \cup \{\infty\}$ into itself by

$$N_f(z) = z - (f(z)/f'(z)).$$

Now for Newton's method: Pick an initial point $z_0 \in \mathbf{C}$ and generate the *orbit*

$$z_0, z_1 = N_f(z_0), z_2 = N_f(z_1), \ldots, z_{k+1} = N_f(z_k) = N_f^{k+1}(z_0), \ldots.$$

Some stopping rule such as "stop if $| f(z_k) | < \epsilon$ and output z_k (else pick a new initial point if k is too large)" is implied.[6]

We can represent Newton's method schematically as in Figure 2.

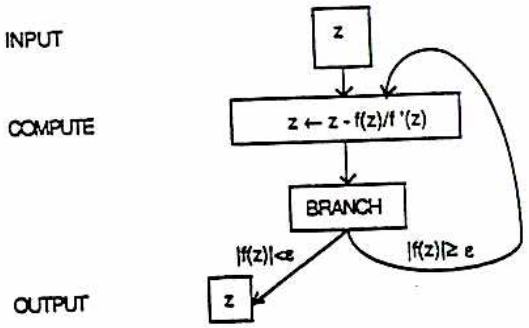

FIGURE 2 The Newton machine for *f*.

A Turing machine for implementing Newton's method, by reducing all operations to bit operations, would wipe out its basic underlying structure. We would like to have a model of computation in which Newton could be represented as naturally as in the Newton machine, and in which its salient features would be as apparent.

3 Does P=NP?

If a problem has a solution that can be easily verified, can such a solution be found quickly? This question is formalized by means of the fundamental open problem of classical (discrete) complexity theory, namely does P=NP? We would like to pose this question within a more general setting, thus perhaps increasing the mathematical tools and perspectives available to tackle it.

[6] Simple calculations show that the zeros of f are the fixed points of N_f (i.e. $f(z) = 0$ if and only if $N_f(z) = z$), and the fixed points of N_f are attracting (i.e. $| N_f'(z) | < 1$). This implies there are local neighborhoods about the zeros of f that contract under (iterates of) N_f. Thus any point $z \in S$ that, under the action of N_f, eventually enters one of these contracting neighborhoods will eventually approach a zero of f. This is the basis for Newton's method. It is well known, however, that Newton's method is not *generally convergent*. The main obstruction to general convergence is the existence of attracting periodic points of period at least 2. (See (Smale 1985) and (Friedman 1989) for estimates on measures for the basin of attraction of the Newton map.)

4 Finite Dimensional Machines over a Ring *R*

Now we describe our formal model of computation over a ring. Let R be an arbitrary ordered commutative ring (or field).

Definition: A finite dimensional *machine M over R* consists of three spaces: *input space* $\overline{I}$, *state space* $\overline{S}$, and *output space* $\overline{O}$ of the form R^l, R^n, R^m respectively, together with a finite directed connected graph with four types of nodes: *input*, *computation*, *branch* and *output*.

The *unique* input node has no incoming edges and only one outgoing edge. All other nodes have (possibly several) incoming edges. Computation nodes have only one outgoing edge, branch nodes exactly two (left and right), and output nodes none. Each node has associated maps:

At the *input node*, there is a linear map I taking points from the input space to the state space.

Each *computation node* has an associated polynomial or rational map $g : R^n \to R^n$ of the state space to itself.

Each *branch node* has an associated polynomial function $h : R^n \to R$ from the state space R^n to the ring R. For a given *state* z in R^n at such a node, branching left or right will depend upon whether or not $h(z) < 0$.

Finally each *output node* has an associated linear map from the state space to the output space.

If R is a field and g is a rational map associated with some computation node, we will assume that previous nodes have tested for the vanishing of the denominators occurring in g and branched away as necessary. (Thus we are assuming that a map associated with a computation node is defined at every input to the node.)

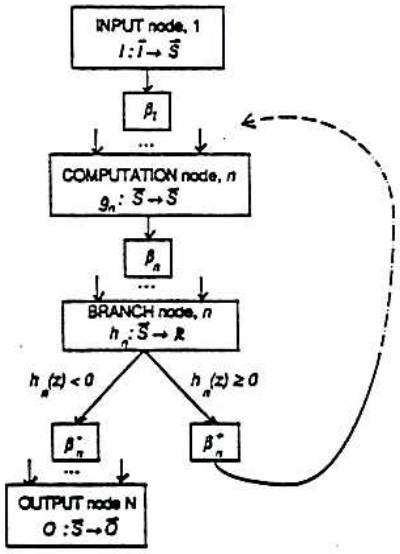

FIGURE 3 A finite dimensional machine *M*: *I* and *O* are the maps associated with the input and output nodes respectively. For *n* a computation node, g_n is the associated "computation map"; and for *n* a branch node, h_n is the associated "branching function.

For *n* an input node or a computation node, β_n is the unique *next node* following *n*. For *n* a branch node, β_n^- is the next node along the left outgoing edge and β_n^+ the next node along the right outgoing edge.

Thus the Newton machine is an example of a machine over $\mathbf{R}$. (Again we are viewing $\mathbf{C}$ as $\mathbf{R}^2$ and the Newton map N_f as a rational map $g = (g_1, g_2) : \mathbf{R}^2 \to \mathbf{R}^2$. By expressing the stopping rule as $|f(z)|^2 < \epsilon^2$, we get an equivalent real polynomial condition $h(x, y) < 0$.)

It is quite natural to view M as a discrete dynamical system. Here it is convenient to assume there is only one output node with associated map O. Thus we may let $\overline{N} = \{1, \ldots, N\}$ be the set of nodes of M, where 1 is the input node and N the output node. We call the space of node/state pairs $\overline{N} \times \overline{S}$ the *full state space* of the machine.

Implicitly associated to M is the *computing endomorphism*

$$H : \overline{N} \times \overline{S} \to \overline{N} \times \overline{S}$$

of the full state space to itself. That is, H maps each node/state pair (n,x) to the unique *next node/next state* pair $(\beta(n, x), g(n, x))$ determined by the directed graph of the machine and its associated maps (see Figure 3.) in the following way:

$\beta(1, x) = \beta_1,\ g(1, x) = x.$

If n is a computation node:

$\beta(n, x) = \beta_n,\ g(n, x) = g_n(x).$

If n is a branch node:

$\beta(n, x) = \begin{cases} \beta_n^- & \text{if } h_n(x) < 0 \\ \beta_n^+ & \text{if } h_n(x) \geq 0 \end{cases}, \quad g(n, x) = x.$

$\beta(N, x) = N,\ g(n, x) = x.$

The computing endomorphism is our main technical as well as conceptual tool. For example, we can use it to define the *input-output map* φ_M of a machine M as follows:

With *input* y in $\overline{I}$, let $x = I(y)$. Then with *initial point* $z_0 = (1, x)$ of the full state space $\overline{N} \times \overline{S}$ generate the *computation* (i.e. the orbit under iterates of H)

$$z_0 = (1, x), z_1 = H(z_0), z_2 = H(z_1), \ldots, z_k = H(z_{k-1}) = (n_k, x_k), \ldots.$$

Halt when (if ever) the first point z_T is produced which has the form $z_T = (N, w)$. If this is the case, the resulting finite sequence is called a *halting computation*; we say M *halts* on input y in *(halting) time* T with *output* $O(w)$ and define $\varphi_M(y) = O(w)$. If there is no such T, then M *does not halt* on input y (i.e. the halting time is infinite) and φ_M is not defined.

The *halting set* of M, Ω_M, is the set of all points in $\overline{I}$ on which M halts. Thus, $\varphi_M : \Omega_M \to \overline{O}$.

The conditions describing halting computations are essentially (semi-)algebraic; they serve as the key technical tool in the proof of the *NP*-Completeness Theorem, as well as in an algebraic proof of Gödel's Theorem (see (Blum and Smale, 1990)). The basic idea is that the relevant sets can be defined in terms of these conditions. For example, the *time* T *halting set* of M can be defined as the set of all points y in $\overline{I}$ for which there are solutions $z_0, \ldots, z_T$ and w to the (time T) *register equations* (of M):

$$z_0 = (1, I(y)), z_T = (N, w) \text{ and } z_k = H(z_{k-1}) \text{ for } k = 1, \ldots, T.$$

Now having defined our formal notion of machine over R, we can easily formalize all related concepts including those in Sections 2 and 3. For example, we define a map

$$\varphi : Y \to R^m,\ Y \subset R^l$$

to be *computable over R* if it is the input-output map of some machine M over R, i.e. if $\varphi = \varphi_M$ and $Y = \Omega_M$. We say M *computes* φ.[7] A set $S \subset R^l$ is *decidable over R* if its characteristic function is computable over R. Otherwise it is *undecidable over R*.

In this setting, Penrose's question may thus be posed quite formally: Is the Mandelbrot set M decidable over **R**?

But before addressing this, it is worth noting that M', the complement of M, is *semi-decidable* over **R**. That is, there is a machine over **R** that on input $x \in \mathbf{R}^2$ outputs 1 if $x \in M'$ and otherwise outputs 0 or is undefined. A semi-decidable machine for M' can be constructed (see Figure 4) using the fact that M is also characterized as $\{c \in \mathbf{C} \mid |p_c^n(0)| \leq 2\}$.

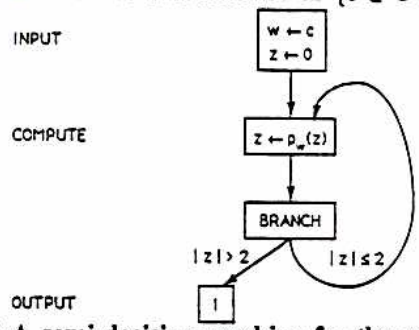

FIGURE 4 A semi-decision machine for the complement of M.

Now as in the classical theory, it is easy to see that a set is decidable just in case both it and its complement are semi-decidable, and that the semi-decidable sets are exactly the halting sets.

Thus we are now ready to take a closer look at halting sets. Here it is convenient to return to the directed graph picture of a machine M over R. To each point y in the halting set Ω_M we associate its *halting path*, i.e. the finite sequence of nodes

$$n_0 = 1, n_1, \ldots, n_T = N$$

traversed from input to output in the *computation* of $\varphi_M(y)$. (The halting path is the sequence obtained by projecting each element of the halting computation with input y onto its first coordinate.)

There are only a countable number of halting paths. For each halting path γ let $\overline{I}\gamma$ be the set of all points in the halting set Ω_M that have γ as their halting path. It is easy to see that for distinct γ's the $\overline{I}\gamma$'s are disjoint. Also, each $\overline{I}\gamma$ is a *semi-algebraic* set.[8] Note also that M

[7] We remark that the new theory reduces to the classical when R=**Z**. That is, the computable functions over **Z** are exactly the recursive functions (see BSS). Therefore, our model of computation is sufficiently powerful to develop the classical theory. (By Church's Thesis, we would have cause for concern had we produced more functions computable over **Z**.)

[8] A set $S \subset R^l$ is *basic semi-algebraic* (over R) if it is the set of elements in R^l that satisfy a (fixed) finite set of polynomial equalities and inequalities (over R). A *semi-algebraic* set is a finite union of basic semi-algebraic sets.

acts like a "straight line program" on $\overline{I}\gamma$. Indeed, by concatenating the input, computation, and output maps that occur along the path γ, we see that φ_M restricted to $\overline{I}\gamma$ is just a polynomial (or rational) map φ_γ. Thus we have the following:

PROPOSITION 1. The halting set of a machine M over R is a (disjoint) countable union of semi-algebraic sets (over R); the input-output map φ_M is a piecewise polynomial (or rational) map.

So, for example, halting sets have integral Hausdorff dimension.

PROPOSITION 2. (Sullivan 1990). The Mandelbrot set is not the countable union of semi-algebraic sets over **R**.[9]

COROLLARY. The Mandelbrot set is not decidable over **R**.

The same holds for most Julia sets since, from the theory of complex analytic dynamical systems, we know that most Julia sets have fractional Hausdorff dimension. Indeed, for hyperbolic rational maps of the Riemann sphere, we have the following

THEOREM (BSS) A Julia set is decidable if and only if it is

1. a round circle,
2. an arc of a round circle, or
3. the whole sphere.

5 Infinite Dimensional Machines over R

The classical construction of a universal machine assumes an effective coding of machines by (natural) numbers. In effect, the coding is a collapsing of sequences of numbers into a single number. Over the integers this can be done by a gödel coding. However, in general over a ring R (e.g. over the reals), such an invertible collapsing cannot be done by a computable map. But even more, this collapsing destroys the algebraic structure of the underlying spaces and so we wish to avoid this approach in our development.

Our resolution (BSS) is reminiscent of the viewpoint, and terminology, used by programmers, i.e. that a program is its own code. To each machine M over R we associate a program $\pi(M)$ which will be a finite sequence of elements of R describing the workings of M. Such a program will be M's code. This suggests that a universal machine over R should have the facility to take as input finite sequences of unbounded length.

In addition, if we wish to have a natural framework for dealing with uniform procedures for solving problem instances of arbitrary dimension (say for the Travelling Salesman Problem), we are also led to consider machines that handle unbounded sequences.

With these considerations in mind we are motivated to extend our notions to *infinite dimensional machines* over R:

The underlying spaces $\overline{I}$, $\overline{S}$, and $\overline{O}$ for an infinite dimensional machine over R each will be R^∞, the *infinite direct sum* space over R. A point $y = (y_1, y_2, \ldots)$ in R^∞ satisfies $y_k = 0$ for k sufficiently large. The *length* of y is the largest n such that $y_n \neq 0$. Polynomial (or rational) maps in this context are still defined by a fixed finite number of polynomials (rational functions) that and depend only on a fixed finite number of variables

[9] Here I would like to acknowledge helpful discussions with Michel Herman and Adrien Douady and also with John Hubbard who provided an independent proof of this result.

The machine will consist of a finite connected directed graph now containing five types of nodes, four as before, with associated maps. If the machine had only the previous type nodes it would essentially be a finite dimensional machine. The increased power comes from the addition of *fifth nodes* that allow accessing of coordinates of arbitrarily high dimension.

A *fifth node* may have several incoming edges but only one outgoing edge. The associated map transforms state $x = (i, j, x_1, \ldots, x_j, \ldots, x_k, \ldots, x_i, \ldots)$ to state $x' = (i, j, x_1, \ldots, x_i, \ldots, x_k, \ldots, x_i, \ldots)$, assuming i and j are positive integers. That is, the fifth node map writes x_i in the "jth place" of x and leaves everything else alone.

Thus, the first two coordinates of the state space play a special role which require some minor modification of the other maps. For $y = (y_1, y_2, \ldots)$ in $\overline{I}$ we let $I(y) = (1, 1, length(y), y_1, 0, y_2, 0, y_3 \ldots)$. This initializes the indices i and j and leaves room for workspace. Information about the length of y is often useful and so is also included. For $x = (i, j, x_1 x_2, \ldots)$ in $\overline{S}$, we suppose a computation node map can alter the first two coordinates only by adding 1 or by setting to 1. Finally we let $O(x) = (x_2, x_4, \ldots)$.

Computing endomorphisms, input-output maps, halting sets and all such related notions are defined exactly as before. Note that a finite dimensional machine can be considered as a special case of infinite dimensional machine.

See BSS for an explicit construction of a universal machine over R.

6 COMPLEXITY THEORY OVER a RING *R*

A goal of computational complexity theory is to quantify the intrinsic difficulty of solving (solvable) problems. This theory had its origins in the 1960's.[10] It was developed primarily by researchers, originally trained in mathematics and logic, but who found more hospitable environments for these interests in the newly emerging computer science departments. (For the

[10] However, as early as 1948, von Neumann (1963) clearly articulated the need for such a theory. From his Hixon Symposium lecture: "Throughout all modern logic, the only thing that is important is whether a result can be achieved in a finite number of elementary steps or not. The size of the number of steps which are required, on the other hand, is hardly ever a concern of formal logic. Any finite sequence of correct steps is, as a matter of principle, as good as any other. It is a matter of no consequence whether the number is small or large, or even so large that it couldn't possibly be carried out in a lifetime, or in the presumptive lifetime of the stellar universe as we know it....[On the other hand] in the case of an automaton the thing which matters is not only whether it can reach a certain result in a finite number of steps at all but also how many such steps are needed." A primary concern here for von Neumann was his conviction that the cumulative effect of the small but non-zero probability of component failure "may (if unchecked) reach the order of magnitude of unity— at which point it produces, in effect, complete unreliability." To the contrary in fact, as far as I am aware, component failure has not posed difficulties anywhere near the magnitude posed by the (apparent) intractability phenomenon.

Much work of logicians since the 1930's had been in identifying and classifying decidable and undecidable problems and theories. With the advent of the digital computer, and its promise of solving hitherto intractable problems, interest perked the realm of the solvable (i.e. decidable) with the quest for efficient algorithms. Although there were many successes, it soon became apparent that a number of problems (such as the Travelling Salesman problem, while solvable in principle, defied efficient solution. These problems seemed in essence intractable. Thus, amongst the solvable, there appeared to be yet another rich and natural hierarchy, with the dichotomy of tractability/intractability mirroring the earlier dichotomy of decidability/undecidability.

seminal work in the theory, see (Rabin 1960), (M. Blum 1967), (Winograd 1970), (Cook, 1971), (Karp, 1972) and (Levin 1972).)

Classical complexity theory deals primarily with combinatorial (discrete, integer) problems. We extend the theory in order to consider a wider class of problems. As has been traditional however, we focus on *decision problems*. These are problems with "yes/no" answers (to questions generally of the form "Does there exist a solution to...?") and are classified as to their difficulty into *classes P, NP* or as being *NP-complete.*

DEFINITION. A *decision problem* over R is a pair (Y, Y_{yes}) with

$$Y_{yes} \subset Y \subset R^{\infty}.$$

Y is the set of *problem instances*, and Y_{yes} is the set of *yes-instances.*

For example the Travelling Salesman Problem, stated over an ordered ring R, can be put in this form by letting:

$$Y = \{(n, A, k) \mid n \text{ is a positive integer, } k > 0 \text{ and } A = (a_{ij}) \text{ is an } n \times n \text{ matrix over } R\}$$
$$Y_{yes} = \{(n, A, k) \text{ in } Y \mid \text{ there is a } tour\ \tau(n) \text{ with } Distance(A, \tau(n)) \leq k\}.$$

Here a *tour* $\tau(n) = (\tau_1, \tau_2, \ldots, \tau_n)$ is a cycle on the entire set {1,2,...,n} and $Distance(A, \tau(n)) = \left(\sum_{i=1}^{n-1} a_{\tau_i \tau_{i+1}}\right) + a_{\tau_n \tau_1}$. By representing A by the sequence of its rows one after the other, we have $Y \subset R^{\infty}$.

A second example, which will be prominent in out theory, is the *4–Feasibility Problem (4–FEAS)* over R. Here,

$$Y = \{ \text{ multivariable polynomials } f \text{ over } R \mid degree\ f \leq 4\}$$
$$Y_{yes} = \{f \text{ in } F \mid f(\xi) = 0 \text{ for some } \xi = (\xi_1, \ldots, \xi_k) \text{ in } R^k\}$$

We are supposing that polynomials are represented as elements of R^{∞} via the *standard representation* (see BSS).

Thus the 4–FEAS problem is: Given a multivariable polynomial f of degree 4 with coefficients from R, does $f(x) = 0$ have a solution over R? While it may not at all be obvious how to decide if such a solution exists, it is a straightforward procedure to verify one that may be presented to us. Just plug the purported solution into the equation and check it out. Is this verification tractable in our model of computation? The answer will depend on the underlying mathematical properties of the ring or field, as well as our measure of complexity. But first we must formalize the basic concepts of size and cost.

We first suppose we have a function *height* defined on R with values in the non-negative reals, e.g. for R=Z or R, and $y \in R$ we might choose *height* (y) to be *logarithmic* height, $log_2(|y| + 1)$, or *unit* height, 1. Then for $y = (y_1, y_2, \ldots, y_n, 0, 0, \ldots) \in R^{\infty}$, we define

$$size(y) = length(y) + height(y)$$

where $height(y) = max\ height(y_i)$. Thus with unit height, *size* reflects the "dimension" of input, whereas over the integers with logarithmic height size reflects the traditional bit length.

For the remainder of this paper, unless otherwise stated, we will suppose unit height for R and logarithmic height for Z.

Now suppose M is a machine over R with a height function defined. Then

$$cost_M(y) = T_M(y) \times h_{max}(y)$$

where $T_M(y)$ is the halting time of M on input y (which may be finite or infinite depending on whether or not y is in the halting set of M) and $h_{max}(y)$ is the maximum height of any element occurring in the computation of M on input y. Over the reals, the cost reflects the number of basic algebraic operations, whereas over the integers, the cost reflects the number of bit operations.

The following definitions make sense only in case height has been defined over R.

DEFINITION. A map φ on (*admissible*) inputs $Y \subset R^\infty$ is *polynomial time computable over R* if there is a machine M over R that computes φ and

$$cost_M(y) \leq poly(size(y)), \text{ for all } y \text{ in } Y.$$

Here *poly* is some polynomial with nonnegative integer coefficients. Polynomial time is meant to formalize our notion of tractability.

Now we are in a position to formally define class P and class NP over R. While the first definition is straightforward, the second is considerably more subtle.

DEFINITION. A decision problem (Y, Y_{yes}) is in *class P (polynomial time)* over R if the characteristic function of Y_{yes} in Y is polynomial time computable over R.

DEFINITION. (Y, Y_{yes}) is in *class NP (non-deterministic polynomial time)* if there is a machine M' which takes as input pairs (y,w) (where y in Y is a problem instance and w in R^∞ is thought of as a "guess" or "witness" for a solution to y), outputs 1 or 0 (yes or no) and satisfies:

1. If y is a yes-instance then there exists some (guess for a solution) w such that $\varphi_{M'}(y, w) = 1$ and
$$cost_{M'}(y, w) \leq poly(size(y)).$$
2. If y is a *no-instance* (i.e. not a yes-instance) then there is *no* (guess) w such that $\varphi_{M'}(y, w) = 1$.

We remark that we need only consider guesses w for which $size(w) \leq poly(size(y))$.

M' is called an *NP-decision machine* for the *NP-problem* (Y, Y_{yes}). Property 1 reflects the non-deterministic aspect of this notion, i.e. for each yes-instance, we just require that some polynomial time verifiable solution exists, not necessarily that one can be found. Property 2 requires that the verification process have some integrity, i.e. it can never output yes for a no-instance input.

In this general setting it is natural to ask (analogous to the classical question over Z): Does *P=NP* over R? For R=**R**, we have a new **open problem.**

Now let us return to 4–FEAS. The halting time for verifying a purported solution to a polynomial equation $f(x) = 0$ using a straightforward evaluation process can easily be seen to be bounded above by a polynomial function of the length of f. (Recall we are supposing f is standardly represented as an element of R^∞.) Thus over the reals, since size is length and cost is halting time, this verification is polynomial time.

On the other hand, over the integers we know (by the undecidability of Hilbert's Tenth Problem) that even the *smallest* size of integer solutions to polynomial equations $f(x) = 0$ (solvable over **Z**) cannot be bounded above by any polynomial (in $size(f)$). Thus, even if we only consider solutions of smallest size, there is no polynomial that will bound the cost of verification; in general it would just take too long to even read in purported solutions.

The above arguments show that 4–FEAS is in class *NP* over **R** but not over **Z**.

A key impetus for the development of classical complexity theory was the discovery (by Cook (1971) and Levin (1973)) of the existence of *NP* problems (over **Z**) that efficiently encode all *NP* problems.

DEFINITION. $\left(\widehat{Y}, \widehat{Y}_{yes}\right)$ is *NP-complete* if it is in class *NP* and *universal* in the following sense:

For every (Y, Y_{yes}) in *NP* there is a polynomial time map $\varphi : Y \rightarrow \widehat{Y}$ such that for all y in Y

$$y \text{ is in } Y_{yes} \text{ if and only if } \varphi(y) \text{ is in } \widehat{Y}_{yes}.$$

Here φ is the efficient (i.e . polynomial-time) *coding* function. Thus any decision procedure for $\left(\widehat{Y}, \widehat{Y}_{yes}\right)$ can be easily converted (in polynomial time) into one for (Y, Y_{yes}) of not worse complexity (up to a polynomial): To decide if y is in Y_{yes}, simply encode y into $\widehat{Y}$ using a polynomial time machine for φ and then decide if $\varphi(y)$ is in $\widehat{Y}_{yes}$.

Thus an *NP*-complete problem is the "hardest" problem in the class *NP*; any *NP* problem can be efficiently "reduced" to it.

We have the following analogue over **R**, to the pivotal Cook Theorem (3–SAT is *NP*-complete)[11] over **Z**:

MAIN THEOREM (BSS). The 4–Feasibility Problem (4–FEAS) is *NP*-complete over the reals.

REMARKS. This theorem has a number of immediate consequences which point to the subtle differences between the theory of *NP* over the integers and over the reals.

For example, over the integers it is easy to see, using a simple counting argument, that *NP* problems are decidable in exponential time (in the size of the instance). This is because, as noted earlier, for problem instances y, we need only consider guesses of size at most $poly(size(y))$. Over **Z**, there are at most $2^{poly(size(y))}$ such guesses, and so a perfectly good decision procedure is to check out each one in turn using an *NP*-decision machine for the problem.

On the other hand, over **R** there are a continuum number of such guesses, and so it is not even clear that *NP* problems are decidable over **R**, no less decidable in exponential time. However, by Tarski (1951), 4–FEAS is decidable over **R**. So by the *NP*-completeness of 4–FEAS we see that all *NP* problems are decidable over **R**. Moreover, (by Canny (1988) and Renegar (1988)) 4–FEAS is decidable in exponential time (over **R**),[12] and so all *NP* problems must be decidable

[11] Cook's *Satisfiability Problem* is: Given a Boolean formula $\varphi(u_1, \ldots, u_k)$ is there an assignment to the variables $u_1, \ldots, u_k$ that makes the formula true? For 3–SAT the Boolean formulas considered are conjunctions of clauses of the form "U or V or W". Here each of U, V or W is either a variable or the negation of a variable.

[12] For related results with respect to bit complexity see (Grigor'ev and Vorobjov 1988).

in exponential time. Thus we have the same result here over the reals as over the integers but now for much deeper reasons.

The Main Theorem implies that the *P=NP*? problem over **R** is equivalent to the new **open problem:** Is 4–FEAS in class *P* over **R**? (thus focusing our attention on an intrinsic algebraic-geometric problem new to complexity theory.) In contrast, recall over the integers, 4–FEAS is not *even* decidable over **Z.**

The analogous *NP*-complete problem over the complex numbers **C** is related to an effective version of Hilbert's Nullstellensatz.[13] Thus, as in the case of the reals, the *NP*-complete problem here is of a fundamental nature. However, for the moment, these are essentially the only *NP*-complete problems known over the reals or complex numbers.

To contrast, what makes the classical theory of *NP*-completeness so compelling has been the discovery (indicated first by the work of (Karp 1972)) of a large number of seemingly unrelated *NP*-complete problems. A polynomial time decision method for one would yield polynomial time decision methods for all.

Open problem: Find other (seemingly unrelated) *NP*-complete problems over the reals or complex numbers.

The TSP is *NP*-complete over **Z** and, as remarked earlier, in class *NP* over **R.**

Open problem: Is TSP *NP*-complete over **R**?

Proof of MAIN THEOREM (Idea). The task at hand is to show, for each *NP*-problem (Y, Y_{yes}) over **R**, how to encode in polynomial time any problem instance **y** as a degree 4 polynomial $\widehat{\mathbf{y}}$ over **R** such that:

$$\mathbf{y} \text{ is a } yes-instance \text{ if and only if } \widehat{\mathbf{y}} = 0 \text{ has a solution over } \mathbf{R}.$$

The basic idea is to utilize the register equations for an *NP*-decision machine for (Y, Y_{yes}).

First suppose M is any machine over R. Note that the assertion " M with input y outputs x in time T " is equivalent to asserting the system of equations

$$z_0 = (1, I(\mathbf{y})), z_{\mathbf{T}} = (N, x_{\mathbf{T}}), O(x_{\mathbf{T}}) = \mathbf{x} \text{ and } z_k = H(z_{k-1}) \text{ for } k = 1, \ldots, \mathbf{T}.$$

is solvable over R. We can convert (in polynomial time) this essentially semi-algebraic system over **R** to a single polynomial equation of degree 4

$$f(\mathbf{y}, \mathbf{x}, u_1, \ldots, u_{\mathbf{T}'}) = 0$$

such that the original system is solvable over **R** if and only if the single equation is. Here $\mathbf{T}' = p(\mathbf{T})$ where p is some polynomial dependent only on M. See BSS for details.

---

[13] A machine over **C** is similar to one over **R** except at branching nodes; over **C** branching left of right will depend on whether or not a polynomial h evaluated at the current state x is equal to 0. The *NP*-complete problem over **C** is: Given a system $f_1, \ldots, f_k$ of polynomials in n variables $x_1, \ldots, x_n$ over **C**, decide if there is a common solution over **C**. By Hilbert's Nullstellensatz, $f_1, \ldots, f_k$ has no common solution just in case 1 is in the ideal generated by $f_1, \ldots, f_k$, i.e. if and only if $1 = a_1 f_1 + \ldots + a_k f_k$ for some polynomials $a_1, \ldots, a_k$ over **C** in the variables $x_1, \ldots, x_n$.

Now suppose M is an *NP*-decision machine for (Y, Y_{yes}) with time bound a polynomial q. For $\mathbf{y}$ in Y let $\mathbf{T} = q(size\,\mathbf{y})$. Suppose $\mathbf{w}$ is in $\mathbf{R}^\infty$ and $size(\mathbf{w}) = \mathbf{T}$. By the above we have:

"M with input $(\mathbf{y}, \mathbf{w})$ halts with output 1 in time $\mathbf{T}$ " if and only if there is a solution to $f((\mathbf{y}, \mathbf{w}), 1, u_1, \ldots, u_{\mathbf{T}'}) = 0$.Here $\mathbf{T}' = p(\mathbf{T}) = p(q(size\,\mathbf{y}))$.

Now we are ready to encode: For each y in Y let $\widehat{\mathbf{y}}$ be the degree 4 polynomial $f((\mathbf{y}, w), 1, u_1, \ldots, u_{\mathbf{T}'})$ as above (having constant $\mathbf{y}$ and $\mathbf{T} + \mathbf{T}'$ variables $w = (w_1, \ldots, w_{\mathbf{T}})$ and $u_1, \ldots, u_{\mathbf{T}'}$). This is a polynomial time encoding over $\mathbf{R}$.

Now by the definition of *NP*-problems and *NP*-decision machines, y is in Y_{yes} if and only if: there is a w in $\mathbf{R}^\infty$ with $size(w) = \mathbf{T}$ such that M with input $(\mathbf{y}, w)$ halts with output 1 in time T. By the above, this holds if and only if: there is a solution over $\mathbf{R}$ to $f((\mathbf{y}, w), 1, u_1, \ldots, u_{\mathbf{T}'}) = 0$, i.e. to $\widehat{\mathbf{y}} = 0$.

7 Conclusion and Directions

Since this framework is new there are a number of open problems, some of which have already been indicated and new directions to take. Many questions naturally arise concerning the relationship between the classical and new fundamental "*P=NP*?" questions, and even whether the various possible extensions of the classical notion of *NP* (e.g. via guesses or via non-deterministic computation) are equivalent. (This is related to the question of whether or not the TSP is *NP*-complete over $\mathbf{R}$.) Here it is proving fruitful to investigate the fundamental question under varying assumptions on height, computing power and branching criteria. (See (Shub 1990b).) In the general setting, algebraic topology is providing useful tools for lower bound arguments on the topological complexity (i.e. branching complexity) of problems (Smale 1987; Vasiliev 1988; Levine 1989; Hirsch 1990). Related questions are also being pursued over other fields e.g. the p-adics (Bishop 1990).

Another direction is to study questions of parallel and distributed computation, as well as probabilistic algorithms, in this context. For the latter it would be natural to add "coin tossing" nodes to machines. Related to distributed computation, Luo and Tsitstklis (1990) have recently given tight lower bounds for the communication complexity of several algebraic problems.

To bring the theory closer to numerical analysis and scientific computation one must extend the new model of computation to incorporate notions of round-off errors, condition numbers and approximate solutions. See (Renegar 1990) and (Priest 1990). Here questions of the relationship between the complexity and the condition of a problem arise. In this direction it would also seem natural to adjoin nodes to compute limits of (rapidly) converging sequences, as well as other reasonable functions.

Finally there are interconnections between logic (and computation/complexity theory) and the theory of complex analytic dynamical systems to pursue. This is an intriguing direction. For example, inspired by the "degree theory" of classical recursive function theory, one is led to study the hierarchy of Julia sets imposed by various notions of relative decidability. Roughly we say a set A is *decidable relative to* a set B ($A \leq B$) if a machine with an additional node for deciding B (i.e. an "oracle" for B) can be used to decide A.The question then is: what is the resulting hierarchy? Classically, it was an open problem for a number of years (a variant of Post's problem) to find two semi-decidable sets of integers that were incomparable with respect

to relative decidability. (See (Rogers 1967).) Over **R**, Chong (1990) has shown that the situation appears to be quite the opposite, at least for undecidable Julia sets of quadratic maps.[14] Thus we ask: are there two comparable undecidable Julia sets? Alternatively, is there a natural way to increase the power of machines so that the resulting hierarchy is meaningful?

In the opposite direction we have exploited the analog between computing machines and dynamical systems in our *NP*-completeness proofs over the reals (Blum, Shub and Smale 1989) and for a new proof of Gödel's Theorem (Blum and Smale 1990). Here the computing endomorphism is our key technical tool. Can we exploit this analogy further and use techniques of dynamical systems to better understand the nature of complexity of computations and of formal computing machines? In concrete form, this approach has been successfully used by Batterson (1990) (following Shub (1983) and Smale (1985)) for the global analysis of classical algorithms of numerical linear algebra.

[14] Technically, we are talking about the complements of Julia sets which are semi-decidable over **R**.

REFERENCES

Aho, A., Hopcroft, J., and J. Ullman. (1979): *The Design and Analysis of Computer Algorithms*. Reading, Mass: Addison-Wesley

Batterson, S. (1990): "Dynamics of Eigenvalue Computation." To appear, *Proceedings of the Smalefest*

Bishop, E. (1990): "Computation and Complexity over Q_p." Work in progress

Blum, L. (1990): "Lectures on a Theory of Computation and Complexity over the Reals (or an Arbitrary Ring)." *Lectures in the Sciences of Complexity II* (ed. E. Jen), Addison Wesley, 1–47 and TR-89-065, International Computer Science Institute, December 1989

Blum, L. and M. Shub. (1986): "Evaluating Rational Functions: Infinite Precision is Finite cost and Tractable on Average." *SIAM Journal of Computing* 15:2, 384-398

Blum, L., Shub, M. and S. Smale. (1989): "On a Theory of Computation and Complexity over the Real Numbers: *NP*-Completeness, Recursive Functions and Universal Machines." *The Bulletin of the American Mathematical Society*, Vol. 21, No.1, 1-46

Blum, L. and S. Smale, (1990): "The Gödel Incompleteness Theorem and Decidability over a Ring."To appear, *Proceedings of the Smalefest*

Blum, M. (1967): "A Machine-Independent Theory of the Complexity of Recursive Functions." *Journal of the Assoc. for Computing Machinery*, 14, 322–336

Branner, B. (1989): "The Mandelbrot Set." *Chaos and Fractals, Proceedings of Symposia in Applied Mathematics (AMS)*, 39, 75–105

Canny, J. (1988): "Some Algebraic and Geometric Computations in PSPACE." *Proceedings of the 20th Annual ACM Symposium on the Theory of Computing*. 460–467.

Chong, C.T. (1990): "Reducibility of Julia Sets in Complex Analytic Dynamics." preprint

Church, A. (1936): "An Unsolvable Problem of Elementary Number Theory." *American Journal of Mathematics*, 58, 345-363.

Cook, S.A. (1971): "The Complexity of Theorem Proving Procedures." In *Proceedings 3rd ACM STOC*, 151-158.

Cutland, N.J. (1980): *Computability, An Introduction to Recursive Function Theory*. Cambridge, England: Cambridge University Press

Davis, M. (1982): *Computability and Unsolvability*. New York: Dover Publications Inc.

Davis, M., Matijasevic, Y., and J. Robinson, (1976): "Hilbert's Tenth Problem. Diophantine Equations: Positive Aspects of a Negative Solution". In *Mathematical Development Arising from Hilbert's Problems*, edited by F.E. Browder, Proceedings of Symposia in Pure Mathematics, 28: part 1 and part 2. Providence: Amer. Math. Soc., 323-378

Douady, A. and J. Hubbard, (1984, 1985): "Etude Dynamique des Polynomes Complex, I, 84–20, 1984 and II, 85–40. 1985, Publ. Math. d'Orsay, Univ. de Paris-Sud, Dept. de Math. Orsay, France

Friedman, H. and K. Ko. (1982): "Computational Complexity of Real Functions." *J. Theoret. Comput. Sci.* 20, 323–352

Friedman, H. and R. Mansfield. (1988): "Algorithmic Procedures." Preprint, Penn State

Friedman, J. (1989): "On the Convergence of Newton's Method." *Journal of Complexity*, **5**, 12–33

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

Gödel, K. (1931): "Uber formal Unentscheidbare Satze der Principia Mathematica und verwanter System I." *Monatshefte fur Mathematik und Physik* **38**, 173-198

Grigor'ev, D. Yu. and N. N. Vorobjov. (1988): "Solving Systems of Polynomial Inequalities in Subexponential Time," *Journal of Symbolic Computation* **5**, 37–74

Heintz, J, M.-F. Roy and P. Solerno, (1989): "Sur la Complexite du Principe de Tarski-Seidenberg." Preprint

Hilbert, D. (1901–1902): "Mathematical Problems." *Bull. of the Amer. Math. Soc.* **8**, 437–479

Hirsch, M. (1990): "Applications of Topology to Lower Bound Estimates in Computer Science," PhD thesis, UC Berkeley

Karp, R. (1972): "Reducibility among Combinatorial Problems." *Complexity of Computer Computations*, edited by R. Miller and J. Thatcher, New York: Plenum Press, 85–104

Kleene, S.C. (1936): "General Recursive Functions of Natural Numbers." *Mathematische Annalen*, vol. 112, 727–742

Levin, L.A. (1973): "Universal Sorting Problems." *Problemy Peredaci Informacii* **9**, 115–116 (in Russian). English translation in *Problems of Information Transmission* **9**, 265–266

Levine, H. (1989): " A Lower Bound for the Topological Complexity of Poly(d, n). *Journal of Complexity* **5**, 34–44

Luo, Z-Q., and J.N. Tsitsiklis. (1990): "Communication Complexity of Algebraic Computation (Extended Abstract.)."*Proceedings of the 31st Annual Symposium on Foundations of Computer Science*, IEEE Computer Society Press, 758–765

Manna, Z. (1974): *Mathematical Theory of Computation.* New York: McGraw Hill

Matijasevich, Y. (1970): "Enumerable sets are Diophantine." *Soviet Math. Doklady*, **11**, 354–357

Michaux, C. (1990): "Ordered Rings over which Output Sets are Recursively Enumerable Sets." preprint, Universite de Mons, Belgium

Minsky, M. (1967): *Computation: Finite and Infinite Machines.* Englewood Cliffs, NJ: Prentice-Hall

Penrose, R. (1989): *The Emperor's New Mind.* Oxford University Press

Post, E. L. (1936): "Finite Combinatory Processes. Formulation I." *Journal of Symbolic Logic*, vol.1, 103–105

Pour-El, M.B., and I. Richards. (1983): "Computability and Noncomputability in Classical Analysis." *Trans. Amer. Math. Soc.* **275**, 539-560

Priest, D. (1990): "Precision Analysis: An Approach to Designing Accurate Computations in Floating Point Arithmetic." preprint

Rabin, M.O. (1969): "Degree of Difficulty of Computing a Function and a Partial Ordering of the Recursive Sets." Technical Report No. 2, Hebrew University, Jerusalem

Renegar, J. (1988): "A Faster PSPACE Algorithm for Deciding the Existential Theory of the Reals." *Proceedings of the 29th Annual Symposium of Computer Science.* IEEE Computer Society Press, 291-295

Renegar, J. (1989)": "On the Computational Complexity and Geometry of the First-Order Theory of the Reals, Parts I, II and III." Technical Report nos. 853, 854, 856, School of Operations Research and Industrial Engineering, Cornell University, Ithaca

Renegar, J. (1990): "Towards a General Theory of Condition Numbers for Algebraic Problems." To appear, *Proceedings of the Smalefest*

Rogers, Jr., H. (1967): *Theory of Recursive Functions and Effective Computability.* New York: McGraw Hill

Rosser, B. (1936): "Extensions of Some Theorems of Gödel and Church." *Journal of Symbolic Logic* 1, 87-91.

Shub, M. (1983): "The Geometry and Topology of Dynamical Systems and Algorithms for Numerical Problems." Lectures at Beijing University, Beijing, China

Shub. M. (1990a)"Some Remarks on Bezout's Theorem and Complexity Theory," IBM Research Report # 71048, 1990 and to appear, *Proceedings of the Smalefest*

Shub, M. (1990b): "On the Work of Steve Smale on the Theory of Computation." IBM Research Report #71437, 1990 and to appear, *Proceedings of the Smalefest*

Smale, S. (1985): "On the Efficiency of Algorithms of Analysis." *Bulletin of the Amer. Math. Soc.* 13:2, 87–121

Smale, S. (1987): "On the Topology of Algorithms I." *Jour. of Complexity* 3 81-89.

Smale, S. (1990): "Some Remarks on the Foundations of Numerical Analysis." *SIAM Review*, vol. 32, No. 2, 211–220

Sullivan, D. (1990): personal communication

Tarski, A. (1951): *A Decision Method for Elementary Algebra and Geometry.* 2nd revised ed. Berkeley, California: University of California Press, 1951. (The main results of this paper were discovered by Tarsi in1930 and first mentioned in print a year later ("Sur les ensembles definissables de nombres reels I." *Fundamenta Mathematicae*, vol. 17, 210–239)

Turing, A.M. (1937): " On Computable Numbers, with an Application to the Entscheidungsproblem." *Proc. Lond. Math.Soc.* (ser.2), 42, 230–265.

Vasiliev, V. (1988) "Cohomology of the Braid Group and the Complexity of Algorithms." Preprint and to appear *Proceedings of the Smalefest.*

Von Neumann, J. (1963): "The General and Logical Theory of Automata." *Collected Works, Vol V.* (A. Taub, Ed.) MacMillan, New York, 288–328.

Von zur Gathen, J. (1988): "Algebraic Complexity Theory." Technical Report No. 207/88, Department of Computer Science, University of Toronto, 37pp.

Whitehead, A. N. and B. Russell. (1910, 1912, 1913)): *Principia Mathematica*, vol.1 (1910), Vol. 2 (1912). vol. 3 (1913), Cambridge University Press.

Winograd, S. (1970): "On the Number of Multiplications Necessary to Compute Certain Functions." *Comm. Pure and Appl. Math* vol. 23, 65–79.

Winograd, S. (1980): "Arithmetic Complexity of Computations." *SIAM Regional Conference Series in Applied Mathematics* 33, 93 pp.

Yao, A. (1989): "Lower Bounds for Algebraic Computation Trees with Integer Inputs." *Proceedings of 1989 FOCS.* 308–313