COMPUTABILITY AND COMPLEXITY

Martin Davis
Professor Emeritus, Courant Institute, New York University, and Visiting Scholar, Mathematics Department, University of California, Berkeley ,USA

Keywords: arithmetical hierarchy; Bounded Quantifier Theorem; Chinese Remainder Theorem; Church-Turing thesis; complexity measure; computable function; computable set; creative set; Diophantine set; Enumeration Theorem; Fibonacci number; halting problem; Hilbert's 10th Problem; m-complete set; maximal set; MRDP theorem; *NP* - complete problem; oracle; $P = NP$ problem; partially computable function; polynomial-time computability; priority method; productions; r.e. degree; r.e. set; recursive function; recursive set; recursively enumerable set; simple set; true arithmetic; Turing degree; universal Diophantine equation; unsolvable problem; word problems.

Contents

Summary

Computability theory begins with a precise explication (known as the Church-Turing Thesis) of what it means to say that a problem is solvable by an algorithm. This explication can be given in a number of different ways, the most popular of which involve Turing machines. The methods of computability theory make it possible to study problems that can not be solved in this manner, that are as one says, not computable.

A recursively enumerable (r.e.) set of natural numbers is the set of natural number

inputs to some given Turing machine which result in the machine eventually halting. Although computable sets are r.e., there are also r.e. sets that are not computable. Moreover such sets come in a variety of kinds, and these lead to particular forms of Gödel's Incompleteness Theorem.

Computability theory makes it possible to prove that problems in various branches of mathematics fail to have algorithmic solutions. We survey problems in algebra (word problems) and number theory (Hilbert's tenth problem) that have been proved unsolvable in this sense.

Using computability theory one can make sense of assertions to the effect that one problem is more unsolvable than another. Classification of problems from this point of view is seen to have important connections with classifications based on the number of logical quantifiers needed to specify a particular problem.

Computability theory is based on an idealization: one is dealing with computation without limits to available resources of space and time. Complexity theory attempts a somewhat more realistic approach in which resource bounds are explicitly taken into account. There is a large class of problems for which the best available algorithms involve a systematic time-consuming search. The "millennium" *P=NP* problem can be thought of as asking whether such searches can be avoided, whether there are shortcuts.

1. Introduction

Familiarity is assumed with the discussion of Turing machines, computability, the Church-Turing thesis, and unsolvable problems in the chapter on *Formal Logic*. In particular, let us consider a function f defined on some *subset* S of the natural numbers $N = 0,1,2,\ldots$ and with natural number values; f is then *undefined* for numbers in $\overline{S} = N - S$, the *complement* of S, i.e., the set of natural numbers that do not belong to S. Such a function is called *partially computable* if there is a Turing machine which started with a coded representation of the number $n \in S$ (e.g., by a string of $n+1$ tally marks $|$) on its tape, will eventually halt with a string encoding of $f(n)$ on its tape, whereas if it is started with a coded version of a number $n \in \overline{S}$, the machine will never halt. In the special case when f is defined on all of N, we call f a *computable function*, and also say that f is *total*. In the future, language will be abused as follows: we will speak of the number n being on a Turing machine tape to mean that it is the string that encodes n that is actually on the tape.

We can also speak of a partially computable function g of any n-tuple of natural number arguments $x_1, x_2, \ldots, x_n$ by beginning instead with such an n-tuple on the tape of a Turing machine. These arguments can be encoded, for example, by representing each x_i by a string of $x_i + 1$ tally marks and then using a single blank square ␣ to separate the adjacent arguments. So for example the arguments $3,1,4$ would be represented by the string

|||| ␣ || ␣ |||||

Again in the special case where g is defined for all natural number values of its arguments, g is said to be *computable.* For historical reasons, computable functions are also referred to as *recursive functions,* and the entire subject dealing with diverse aspects of computability and non-computability is variously referred to as *computability theory, recursive function theory,* and *recursion theory*.

The availability of a precise mathematical characterization of what can be computed opens up to serious study the non-computable. In this brief chapter, we begin, staying close to the computable, with the so-called *recursively enumerable sets.* Even here we find an interesting distinction among varieties of non-computability. We go on to make contact with problems in ordinary mathematics that can now be shown to be algorithmically unsolvable. Next, we go behind the recursively enumerable sets to survey the further reaches of the non-computable. Finally, descending back to the real world of finite resources, we briefly examine the complexity of computation.

2. Recursive and Recursively Enumerable Sets

With each set S of natural numbers we associate its characteristic function C_S defined by:

$$C_S(n) = \begin{cases} 1 & \text{if } n \in S \\ 0 & \text{otherwise.} \end{cases}$$

Then the set S is called *computable* or *recursive* if its characteristic function C_S is computable. Intuitively to say that a set S is computable is to assert the existence of an algorithm that can distinguish members from non-members of S.

Closely related to the notion of recursive set is that of *recursively enumerable* (abbreviated "r.e.") *set*. A non-empty set S is *r.e.* if there is a computable function f such that S is the set of all values assumed by f, i.e., such that

$$S = \{ f(n) : n = 0,1,2,..\}$$

i.e., the recursive function f "enumerates" S. The empty set $\varnothing$ is also taken to be r.e. An alternative, equivalent, characterization of r.e. sets is as follows: S is r.e. if and only if there is a Turing machine such that when started with the code for a natural number n on its tape, the machine will eventually halt if and only if $n \in S$, or equivalently, if there is a partially computable function f which is defined precisely on members of S. Still another characterization (equivalent to the others mentioned) is: S is r.e. if and only if there is a partially computable function f for which: S is the set of all numbers m such that for some n, $f(n)$ is defined and $f(n) = m$.

The relation between the two notions: *recursive set* and *r.e. set* is given by the following:

Theorem. *The set S of natural numbers is recursive if and only if the sets S and $\overline{S}$ are both r.e.*

That recursive sets and their complements are r.e. is pretty obvious. On the other hand if we have computable functions f, g enumerating S and $\overline{S}$, respectively, then an algorithm for computing C_S is as follows:

1. Given natural number n
2. Successively compute
   $f(0), g(0), f(1), g(1), f(2), g(2), \ldots$
   until either:
3. a value $f(k) = n$ is reached, in which case the value $C_S(n) = 1$ is returned,
   or
4. a value $g(k) = n$ is reached, in which case the value $C_S(n) = 0$ is returned.

That either step 3 or step 4 will eventually be reached is clear because every natural number belongs either to S or to $\overline{S}$.

Using the fact that there is a universal Turing machine (as explained in *Formal Logic*) it is easy to prove the following:

Theorem. There is a partially computable function ϕ of two arguments such that for each partially computable function f of one argument, there is a number j such that

$$f(n) = \phi(j, n)$$

where the equality is assumed to mean not only that both sides are equal when defined, but also that, when either side is undefined, the other is as well.

To see this, let the code $\overline{M}$ of Turing machine M be chosen to be a natural number, and indeed let this be done in such a way that every natural number is the code of exactly one Turing machine. Let $\phi(m, n)$ be the value computed by the universal machine U when started with the coded representation of m and some natural number n on its tape, separated by a blank square. Then, if in particular, M is a Turing machine that computes the partially computable function f, then

$\phi(\overline{M}, n) = f(n)$ for all *n*.

Now for each $j \in N$ let W_j be the set of numbers n for which there exists a number m such that $\phi(j, m)$ is defined and $\phi(j, m) = n$. Then we have at once the following

Enumeration Theorem. *For every r.e. set S, there is a number j such that $S = W_j$.*

Defining

$$K = \{n \in N : n \in W_n\}$$

we have the following

Theorem. *K is r.e. but not recursive.*

Since K is the set of all values assumed by the "diagonal" function $\phi(n,n)$, it follows at once that K is r.e. On the other hand if $\bar{K}$ were also r.e., we would have, by the Enumeration Theorem, for some j,

$$\bar{K} = W_j.$$

But this would lead to the contradiction

$$j \in \bar{K} \Leftrightarrow j \in W_j \Leftrightarrow j \in K.$$

If we think of a Turing machine M which eventually halts when started with a number n on its tape if and only if $n \in K$, we see that in proving that K is not computable, we are also demonstrating the unsolvability of the *halting problem* mentioned in *Formal Logic* .

2.1. m-Complete Sets, Creative Sets, and Simple Sets

Given an r.e. set C which is not recursive and an r.e. set $R \subseteq \bar{C}$, since R can't be r.e., there must be numbers $r \in \bar{C} - R$. Such a set C is called *creative* if there is an algorithm enabling one to compute such a number r given the code of a Turing machine M such that R is the set of numbers n for which M will eventually halt when started with n on its tape. More formally:

An r.e. set C is called *creative* if there is a computable function f such that whenever $W_j \subseteq \bar{C}$, we have $f(j) \in \bar{C} - W_j$. Creative sets are obviously not recursive, For, if a creative set C were recursive, then its complement $\bar{C}$ would be r.e., i.e., $\bar{C} = W_j$ for some j. Hence there couldn't be a function f with $f(j) \in \bar{C} - W_j$.

Theorem. *The set K is creative.*

In this case the identity function $f(n) = n$ does the job. Namely, let $W_j \subseteq \bar{K}$. Then for all n,

$$n \in W_j \Rightarrow n \in \bar{K} \Rightarrow n \notin W_n$$

Thus,

$$j \in W_j \Rightarrow j \notin W_j,$$

and so, $j \notin W_j$, and therefore $j \in \bar{K}$. Finally, $j \in \bar{K} - W_j$.

Creative sets, introduced by E.L. Post in 1943, have a number of interesting properties. For example, it can be proved that they are all m-complete in the following sense:
The r.e. set S is called m-*complete* if for every r.e. set R, there is a computable function f such that for all $n \in N$,

$$n \in R \Leftrightarrow f(n) \in S.$$

The converse is also true: namely every m-complete set is creative. In fact, as was shown by Myhill in 1955, creative sets are all *recursively isomorphic* in the following sense:

Sets A, B of natural numbers are said to be *recursively isomorphic* if there is a one-one computable function f onto the natural numbers (a "permutation") such that for all n,

$$n \in A \Leftrightarrow f(n) \in B.$$

Given a creative set C with computable function f as specified in the definition it can be proved that there are infinite r.e. sets U such that $U \subseteq \overline{C}$. The idea is to begin with a number j_0 such that $W_{j_0} = \varnothing$. Then

$$\{f(j_0)\} \subseteq \overline{C} - W_{j_0}.$$

So we find j_1 such that $W_{j_1} = \{f(j_0)\}$. Then

$$\{f(j_0), f(j_1)\} \subseteq \overline{C} - W_{j_1}.$$

So we find j_2 such that $W_{j_2} = \{f(j_0), f(j_1)\}$. Continuing the process one obtains the infinite set

$$U = \{f(j_0), f(j_1), f(j_2), \ldots\}$$

If this process is carried out carefully one can show that U is indeed r.e.

In the light of this result, one is led to ask whether there exist r.e. sets S such that $\overline{S}$ *is infinite but has no infinite r.e. subsets*. In 1943, E.L. Post called sets with this property *simple*, and showed how to construct such sets. Following his ideas we begin with the array shown in Table 1.

	0	1	2	3	…
0	$\phi(0,0)$	$\phi(0,1)$	$\phi(0,2)$	$\phi(0,3)$	…
1	$\phi(1,1)$	$\phi(1,1)$	$\phi(1,2)$	$\phi(1,3)$	…
2	$\phi(2,2)$	$\phi(2,1)$	$\phi(2,2)$	$\phi(2,3)$	…
3	$\phi(3,3)$	$\phi(3,1)$	$\phi(3,2)$	$\phi(3,3)$	…

.	…	…	…	…	…
.	…	…	…	…	…
.	…	…	…	…	…

Table 1: Construction of r. e. *S* simple in the sense of Pest

Now, we imagine the following algorithm by which a set S is defined: Step-by-step this array is traversed, e.g., along the diagonals. As each entry in the array is reached, the computation of the indicated value of ϕ is initiated. However, the traversal is to be interrupted whenever termination of one of these computations occurs. For each value of j, if it should happen that a computation of some $\phi(j,m)$ terminates with $\phi(j,m) = n > 2j$, then the first time this occurs, the number n is placed in the set S.

If this is arranged carefully, the set S will be r.e. Furthermore for any j for which W_j is infinite, by our construction $W_j \cap S \neq \varnothing$. Hence, $\overline{S}$ contains no infinite r.e. set. An easy counting argument shows that at most k of the numbers $0,1,\ldots,2k$ have been placed in S by this construction. So, at least k of these numbers belong to $\overline{S}$. Hence $\overline{S}$ is infinite. Putting this information together, we see that S is simple.

In this proof Post devised a technique that has been used in ever-more elaborate ways to construct r.e. sets with desired properties: one produces an infinite list of *requirements* such that a set meeting those requirements automatically has the property in question. In the case of Post's construction of a simple set S, the requirements were:

- for each j for which W_j is infinite, $S \cap W_j \neq \varnothing$;
- for each k, at most k of the first $2k$ natural numbers are in S.

Then, a construction of S was set up, meeting these requirements, one at a time. In more elaborate constructions, such as those using the so-called *priority method*, requirements once met can be "injured" at a later stage, as long as they are eventually fully satisfied. An example using such methods, far too complicated to discuss in this article, is the existence of an r.e. set M that is *maximal* in the following sense: if $Q \supseteq M$ is r.e., then either $Q - M$ or $N - Q$ is finite, where as usual $N = \{0,1,2,\ldots\}$. Of course such a maximal set is simple.

2.2. Algorithmic View of Gödel Incompleteness

Another point of view regarding Gödel's epochal incompleteness theorem (see *Formal Logic*), is provided by considerations involving algorithms and r.e. sets. We begin by considering, in an extremely general manner, what the minimum is to be expected of a system providing mathematical "proofs." Such a proof should presumably take the form of a string of characters. Moreover, a minimum requirement would seem to be that there be an algorithm for testing whether a purported proof of some given proposition really is what it purports to be. Now we restrict attention to propositions of a particular form, namely propositions that assert that some natural number n belongs to some fixed definite set A. Then, invoking the Church-Turing thesis, one can easily show that *the*

set A_P *of* n *for which the proposition* $n \in A$ *is provable in a given proof system* P *must be an r.e. set.* The system P is called *sound* if $A_P \subseteq A$, i.e. if every statement provable in P is true. Then we can easily prove the following:

Theorem. *Let* R *be an r.e. set that is not recursive, and let* P *be a sound proof system for* $\overline{R}$. *Then there is a number* $k \in \overline{R} - \overline{R}_P$, *i.e., the statement* $k \in \overline{R}$ *is true but not provable in the proof system* P.

Otherwise, it would be the case that $\overline{R} = \overline{R}_P$, and this is impossible because $\overline{R}_P$ is r.e. while $\overline{R}$ is not.

This form of Gödel's incompleteness theorem takes on a particularly striking form when the r.e. set R is taken to be creative or to be simple.

Theorem. *Let* C *be a creative set, and let* P *be any sound proof system for* $\overline{C}$ *with* $W_j = \overline{C}_P$. *Then there is a computable function* f *such that* $f(j) \in \overline{C} - W_j$, *i.e., the statement* $f(j) \in \overline{C}$ *is true but not provable in the proof system* P.

It is this property that Post wished to emphasize when he suggested the term "creative." The point is that not only are there unprovable truths of the form $k \in \overline{C}$, but such truths are obtainable algorithmically from a specification of P. This suggests augmenting a given proof system P by accepting the unprovable truth as a new "axiom." And then this process can be iterated obtaining more and more powerful proof systems. To Post this emphasized the "creative" nature of mathematics, inevitably bursting the bounds of any given proof system.

Theorem. *Let* S *be a simple set, and let* P *be a sound proof system for* $\overline{S}$. *Then the set* $\overline{S} - \overline{S}_P$ *is finite.*

Thus, no matter how "powerful" the proof system P may be, it can never serve to prove more than a finite number of propositions of the form $k \in \overline{S}$. Later we shall see that these propositions are each equivalent to a statement about natural numbers involving nothing more complicated than addition and subtraction.

-
-
-

Bibliography

Boone W.W., Cannonito F.B. and Lyndon R.C. (1973). *Word Problems.* North-Holland. Amsterdam. [A collection of articles on word problems.]

Britton J.L. (1963). The Word Problem. *Annals of Mathematics*. **77**, 16-32. [A proof of the unsolvability of the word problem for groups.]

Cutland N.J. (1980). *Computability: an introduction to recursive function theory.*Oxford University Press. Oxford. [An excellent textbook.]

Davis M.D. (1958) *Computability and Unsolvability.* McGraw-Hill, New York reprinted with an additional appendix, Dover 1983. [One of the first books on computability theory. The Dover reprint contains an exposition of the unsolvability of Hilbert's tenth problem.]

Davis M.D. (1973) Hilbert's Tenth Problem is Unsolvable. *American Mathematical Monthly*. **80**, 233-269. [This exposition of the unsolvability of Hilbert's tenth problem is also reprinted in the Dover reprint of the author's *Computability and Unsolvability*.]

Davis M.D., Matijasevic Y.V. and Robinson J.B. (1976). Hilbert's Tenth Problem: Diophantine Equations: Positive Aspects of a Negative Solution. *Proceedings of Symposia in Pure Mathematics*. **28**, 323-378; reprinted in Feferman S., ed. *The Collected Works of Julia Robinson*. American Mathematical Society, Providence 1996, 269-378. [A survey of applications and generalizations of MRDP.]

Green B. and Tao T.(2005). The primes contain arbitrarily long arithmetic progressions. *Annals of Mathematics*, to appear. [The result that, well after the fact, turned the 1959 Davis-Putnam proof with a gap into an actual proof.]

Higman, G. (1961). Subgroups of Finitely Presented Groups. *Proceedings of the Royal Society, London Series A.* **262**, 455-475. [Gives the proof of Higman's marvelous theorem.]

Matiyasevich, Y.V. (1993). *Hilbert's Tenth Problem.* MIT Press, Cambridge, Massachusetts. [An excellent monograph.]

Biographical Sketch

Martin Davis was born in New York City in 1928. He was a student of Emil L. Post at City College and his doctorate at Princeton in 1950 was under the supervision of Alonzo Church. Davis's book "Computability and Unsolvability" (1958) has been called "one of the few real classics in computer science." He is best known for his pioneering work in automated deduction and for his contributions to the solution of Hilbert's tenth problem for which latter he was awarded the Chauvenet and Lester R. Ford Prizes by the Mathematical Association of America and the Leroy P. Steele Prize by the American Mathematical Society. In 1983 he was a Guggenheim Foundation Fellow. His books have been translated into a number of languages including Russian and Japanese. Davis has been on the faculty of the Courant Institute of Mathematical Sciences of New York University since 1965, was one of the charter members of the Computer Science Department founded in 1969, and is now Professor Emeritus. He is a Visiting Scholar at the University of California, Berkeley.