APPROXIMATION OF FUNCTIONS ON THE INTEGERS

GENE F. ROSE AND JOSEPH S. ULLIAN

How can algorithms be used to analyze nonrecursive functions? This question motivates the present work.

Let us suppose that a particular function, with natural numbers as arguments and values, is known to be completely defined but not recursive. Then by Church's thesis,[1] no algorithm gives the functional value for every argument. In some practical situation, however, where a particular sequence of arguments is of interest, it might suffice to have an "approximating algorithm" that performs as follows when applied to the successive arguments in the sequence: for each argument, the algorithm computes a number; for some arguments, this number may differ from the actual functional value, but after sufficiently many arguments have been processed, the proportion of such cases never exceeds a prescribed real number less than unity. If such an approximating algorithm exists whenever the given sequence of arguments is infinite, nonrepeating and effectively generable, then the given function is in some (conceivably useful) sense susceptible to analysis by mechanical means. Functions of this last kind are the object of our investigation; when the above notions are made precise in § 1, they are called "recursively approximable" functions.

In § 2 it is shown that uncountably many nonrecursive functions are recursively approximable; in § 3, that uncountably many functions are not recursively approximable.[2]

1. A number-theoretic notion of approximation. Given any function f, any partial function φ,[3] and any sequence $x_0, x_1, \cdots$ of natural numbers, let "err (n)" denote the number of natural numbers $i < n$ such that $f(x_i) \neq \varphi(x_i)$. If E is a real number and, for all sufficiently large n, err $(n)/n \leqq E$, then we say that φ *approximates*

Received December 5, 1962.

[1] Cf [**3**].

[2] An analogous notion of approximable function, involving finite sets of arguments rather than sequences, is considered in [**5**], where a function is called "m-in-n-computable" if there is an algorithm that produces at least m correct functional values for every set of n arguments. It was shown that uncountably many functions are not m-in-n computable for any $m > 0$. The existence of nonrecursive m-in-n-computable functions with $m > 0$ was left an open question; an affirmative answer, however, was soon provided by Dana Scott in an unpublished communication.

[3] By "function" we mean, unless otherwise specified, "total singulary function" (in the sense of [**1**] p. xxi). A "partial function" is any singulary function whose domain is a subset of the natural numbers.