## 1. Computability

Recursion theory is, at least in its initial stages, the theory of computability. In particular, the first task of recursion theory is to give a rigorous mathematical definition of computable.

A computation is a process by which we proceed from some initially given objects by means of a fixed set of rules to obtain some final results. The initially given objects are called inputs; the fixed set of rules is called an algorithm; and the final results are called outputs.

We shall always suppose that there is at most one output; for a computation with $k$ outputs can be thought of as $k$ different computations with one output each. On the other hand, we shall allow any finite number of inputs (including zero).

We shall suppose that each algorithm has a fixed number $k$ of inputs. We do not, ever, require that the algorithm give an output when applied to every $k$–tuple of inputs. In particular, for some $k$–tuples of inputs the algorithm may go on computing forever without giving an output.

An algorithm with $k$ inputs computes a function $F$ defined as follows. A $k$–tuple of inputs $x_1,...,x_k$ is in the domain of $F$ iff the algorithm has an output when applied to the inputs $x_1,...,x_k$; in this case, $F(x_1,...,x_k)$ is that output. A function is computable if there is an algorithm which computes it.

As noted, an algorithm is set of rules for proceeding from the inputs to the output. The algorithm must specify precisely and unambiguously what action is to be taken at each step; and this action must be sufficiently mechanical that it can be done by a suitable computer.

It seems very hard to make these ideas precise. We shall therefore proceed in a different way. We shall give a rigorous definition of a class of functions. It will be clear from the definition that every function in the class is computable. After some study of the class, we shall give arguments to show that