every computable function is in the class. If we accept these arguments, we have our rigorous definition of computable.

2. Functions and Relations

We must first decide what inputs and outputs to allow. For the moment, we will take our inputs and outputs to be natural numbers, i.e., non–negative integers. We agree that number means natural number unless otherwise indicated. Lower case Latin letters represent numbers.

We now describe the functions to which the notion of computability applies. Let ω be the set of numbers. For each k, ω^k is the set of k–tuples of numbers. Thus ω^1 is ω, and ω^0 has just one member, the empty tuple. When it is not necessary to specify k, we write $\vec{x}$ for $x_1,\ldots,x_k$.

A k–ary function is a mapping of a subset of ω^k into ω. We agree that a function is always a k–ary function for some k. We use capital Latin letters (usually F, G, and H) for functions.

A k–ary function is total if its domain is all of ω^k. A 0–ary total function is clearly determined by its value at the empty tuple. We identify it with this value, so that a 0–ary total function is just a number. A 1–ary total function is called a real. (This terminology comes from set theory, where reals are often identified with real numbers. It will lead to no confusion, since we never deal with real numbers.)

A common type of algorithm has as output a yes or no answer to some question about the inputs. Since we want our outputs to be numbers, we identify the answer yes with the number 0 and the answer no with the number 1. We now describe the objects computed by such algorithms.

A k–ary relation is a subset of ω^k. We use capital Latin letters (generally P, Q, and R) for relations. If R is a relation, we usually write $R(\vec{x})$ for $\vec{x} \in R$. If R is 2–ary, we may also write $x\ R\ y$ for $R(x,y)$.