Effective enumerability

1) A set is *enumerable* if, and only if, it is the range of a total or partial function on the natural numbers.
2) A set is *effectively enumerable* if, and only if, it is the range of a total or partial effectively computable function on the natural numbers.
3) A function f is *effectively computable* if, and only if, there is a list of instructions giving a step-by-step procedure that will determine in a finite number of steps the value $f(n)$ for any argument n for which f returns a value.
4) How many functions on the natural numbers are there?
   a) Consider just the functions on natural numbers that have either 0 or 1 as their value.
   b) Each such function f determines a set Δ of natural numbers—$n \in \Delta$ iff $f(n) = 1$—and for each set Δ of natural numbers there is such a function f.
   c) But the set of all sets of natural numbers is nondenumerable.
   d) Hence, the set of functions from the set of natural numbers to $\{0,1\}$ is nondenumerable.
5) How many lists of instructions are there? No more than there are sentences in language, and the set of sentences is enumerable.
6) Hence, there are uncomputable functions. In fact, *most* numerical functions are uncomputable.
7) There is an effectively computable function on the natural numbers whose range is the set of valid sentences. If the sentences of our formal language are encoded, this is even a numerical function that delivers the code number of a valid sentence for each natural number.
8) Suppose we enumerate the sentences in our formal language and define a function f such that $f(n) = 1$ if sentence number n is valid and $f(n) = 0$ if sentence number n is invalid. f is not computable (Church's theorem).
9) It follow that there is no effectively computable function on the natural numbers whose range is the set of *invalid* sentences. For if there were, we could compute f by criss-crossing from one list to the other.