Exercises on the Church-Turing thesis and coding

Exercise 1. Suppose that $f(x)$ and $g(x)$ are computable functions. Prove, using Church's thesis, that $h(x)$ defined as follows is computable.

$$h(x) = \begin{cases} x & \text{if } x \in \mathrm{Dom}(f) \cap \mathrm{Dom}(g) \\ \text{undefined} & \text{otherwise} \end{cases}$$

Example solution. By Church's thesis, it suffices to describe a procedure that, given x, delivers the value of $h(x)$ if defined, and fails to terminate otherwise. We are going to describe such a procedure. Given x, proceed as follows:

1. first, compute $f(x)$;
2. if and when the computation ends, compute $g(x)$;
3. if and when the computation ends, output x.

Let us show that this procedure computes h. If $h(x)$ is defined, then $x \in \mathrm{Dom}(f) \cap \mathrm{Dom}(g)$; this means that both $f(x)$ and $g(x)$ are defined; so, our procedure terminates and returns x, which is the value of $h(x)$, as desired.

If $h(x)$ is not defined, then $x \notin \mathrm{Dom}(f) \cap \mathrm{Dom}(g)$, which means that either $f(x)$ is undefined, or $g(x)$ is. If $f(x)$ is undefined, the first step of our procedure will not terminate. If $f(x)$ is defined, but $g(x)$ is not, then the second step will not terminate. In either case, the procedure will not terminate, as desired.

Exercise 2. Suppose that $f(x)$ is a total computable function. Prove, using Church's thesis, that $h(x)$ defined as follows is computable.

$$h(x) = \begin{cases} 1 & \text{if } x \in \mathrm{Ran}(f) \\ \text{undefined} & \text{otherwise} \end{cases}$$

Example solution. Again, we just have to describe informally a procedure which yields the value of the function, if defined, and fails to terminate otherwise. Here is such a procedure. Given x, act as follows:

1. start with $k = 0$;
2. compute $f(k)$ (this is possible since f is computable);
3. if the result equals x, output 1;

4. otherwise, increment k by 1 and go back to step 2.

Let us show that this procedure computes h. If $h(x)$ is undefined, then $x \notin \mathrm{Ran}(f)$; thus, there is no k such that $f(x) = k$, which means that our procedure never terminates, as desired. On the other hand, if $h(x)$ is defined, then this means that $x \in \mathrm{Ran}(f)$, i.e., that $f(k) = x$ for some k; now, since f is total, the computation of each $f(i)$ terminates; this means that within a finite amount of time spent computing $f(0), \ldots, f(k-1)$, we will come to compute $f(k)$; at that point, we will find that $f(k)$ equals x, and we will output 1, which is the value of $h(x)$.

Exercise 3. Show that any computable function f has infinitely many indices.

Solution. Since the indices for f are in one-to-one correspondence with the URM programs that compute f, we just have to show that there are infinitely many programs that compute f. Let $P^0 = I_1 \ldots I_k$ be a program that computes f (we know that such a program exists, since f is computable). Then P^n also computes f, where P^n is the program in which the instructions $I_1 \ldots I_k$ are followed by n occurrences of the instruction $T(1,1)$. Since the programs $P^n, n \in \mathbb{N}$ are all different from one another, this shows that there are infinitely many programs that compute f, and therefore infinitely many indices for f.