Chapter 6

The Church-Turing Thesis

Let *Eff* denote the intuitive collection of intuitively effective total functions (not a set since not clearly defined). Since it is easy to see how our ad hoc operators yield intuitively effective functions from intuitively effective functions, and since the basic functions are intuitively effective, we have:

$$Tot \subseteq \mathit{Eff}.$$

I should mention, however, that this is not entirely uncontroversial. Some finitists (e.g., Goodstein) have tried to argue that finitism stops with primitive recursion.

The converse is the **Church-Turing thesis:**

$$\mathit{Eff} \subseteq Tot.$$

This is more troublesome. We have defined Tot in a completely ad hoc way by adding a dumb kind of unbounded serial search on top of a dumb, restricted notion of recursion. It would be a miracle if we caught all the intuitively effective, total functions using these two simple-minded ideas.

But perhaps miracles can happen. Since *Eff* is not a mathematically defined concept (that's the whole point of defining Tot!) we can't prove that $Tot = \mathit{Eff}$. But we can try to provide a philosophical or empirical argument for the thesis. The two standard arguments are as follows.

6.0.1 The Amazing Coincidence Argument

No language for defining intuitively effective functions has ever yielded a definition of a function outside of Tot (although many such notations fall short of capturing all of Tot, as we have seen). This is the argument usually quoted in textbooks. It is a kind of physicist's argument, like discovering that wave mechanics and matrix mechanics are the same theory.

The proofs are tedious but you already know more or less how they go. Let X be a given class of functions computed by a given computational formalism.

To show

$$Part \subseteq X,$$

we show that the basic functions are in X and X is closed under the three partial recursive operators. This is usually easy. Conversely, to show

$$X \subseteq Part,$$

we proceed in a more fussy manner as follows.

1. Use finite sequences of numbers to represent in a natural way instantaneous computational states of computations directed by program M.

2. Code these sequences as single numbers using the primitive recursive Gödel coding.

3. Write a primitive recursive function $init(m, \vec{x}) = s$ that converts a list $\vec{x}$ of input arguments into the Gödel code s of the initial computational state of the computation.

4. Then we write a primitive recursive function $transit(s) = s'$ that transforms each state code number into its successor state code number to simulate computations.

5. Finally, we write a primitive recursive relation $Output(s, y) = y$ that determines whether the process has yet halted with output y.

The work involved is not deep and is similar to what we did when we programmed the universal function (Cf. Cutland for *lots* of this).

6.0.2 Turing's Mathematician Simulation Argument

The preceding argument didn't impress either Church or Gödel, as the notations circulating at the time weren't all that different and the inter-simulation arguments were not that deep. Also, the fact that all the analyses "stop" at the same place doesn't necessarily mean that the capabilities of algorithmic mathematics had been exhausted. It might mean only that everyone had found a natural, easily axiomatized subclass of effectively computable functions whose extension requires a much more subtle insight— sort of like the relationship between Newtonian mechanics and quantum mechanics.

What did impress Gödel was Turing's argument based on Turing machines. As you know from many other classes at this university, Turing computations are opaque and ugly. Well, using them wasn't the point. They were the linchpin of Turing's argument that *Eff* $\subseteq Tot$. The argument goes something like this. Consider a mathematician following an algorithm. The algorithm specifies a finite set of rules for modifying scribbles on a notebook in light of the current state of mind of the mathematician. The mathematician may have boundless originality, but the states of mind relevant to following the algorithm are finite and discrete. Also, only finite many distinct notational states can occupy a page of the scratch pad, else a microscope would be required to follow the algorithm (Turing wryly notes that Chinese characters seem to be an attempt to challenge this assumption). Simple reduction arguments turn the scribblings into bits on a linear tape, the mathematician into a finite state automaton (for the purposes of his involvement in the computation) and the algorithm into a set of rules for elementary operations on the tape. So the mathematician is simulated in all relevant respects by a formal Turing machine. Let Tur denote the set of all Turing-computable functions. The argument purports to show (philosophically) that

$$\mathit{Eff} \subseteq Tur.$$

One can now prove mathematically that $Tur = Part$ along the lines described above. Thus

$$Tot = \mathit{Eff}.$$

Here is an interesting autobiographical description of the reception of the two arguments by Stephen C. Kleene. Since Kleene was a student of Church and invented much of computability theory, he was in a fine position to report!

> Church had been speculating, and finally definitely proposed, that the λ-definaable functionss are all the effectively calculable function— ... which I in 1952... called "Church's thesis". When Church proposed [the CT] thesis, I sat down to disprove it by diagonalizing out of the class of the λ-definable functions. But quickly realizing that the diagonalization cannot be done effectively, I became overnight a supporter of the thesis.
>
> Gödel came to the Institute for Advanced Study [at Princeton] in the fall of 1933. According to a ... letter from Church..., Gödel "regarded [the CT] thesis as thoroughly unsatisfactory". Soon thereafter, in his lectures in the spring of 1934, Gödel took a suggestion that had been made to him byHerbrand in a letter in 1931 and modified it to secure effectiveness. The result was what is now known as "Herbrand-Gödel general recursiveness." ...
>
> In a February 15, 1965, letter to Martin Davis, Gödel wrote, "However, I was, at the time of these lectures [1934] not at all convinced that my concept of recursion comprises all possible recursions...".
>
> Church (1936) and I (1936a) published equivalence proofs for Herbrand-Gödel general recursiveness to λ-definability. So, under Church's thesis, ther were now two exact mathematical characterizqations of the intuitive notion of all effectively calculable functions....
>
> The last of the original three equivalent exact definitions of effective calculablity is computability by a Turing machine [1936-37]. ...
>
> For rendering the identification with effective calculability the most plausible— indeed, I believe compelling— Turing computabil-

> ity has the advantage of aiming directly at the goal [i.e., the mathematician simulation argument]....
>
> It seems that only after Turing's formulation appeared did Gödel accept Church'es thesis, which had then become the Church-Turing thesis.[1]

6.0.3 What the thesis doesn't say

Turing's reductive argument does not show that all mechanical computations are Turing computable or that human intelligence is Turing computable. The argument that the mental state may be treated as though it were a member of a discrete, finite space works only because the mind is assumed to be working out a "mind-less" mathematical algorithm. It is only the intuitively effective or algorithmic that is treated by the argument. To extend this to arbitrary mental or mechanical processes is quite another matter. Of course, this didn't prevent A.I. from doing so.

6.1 Arguments "by Church's Thesis"

Granting the Church-Turing thesis, any crisp procedural specification entitles us to infer that a partial recursive index exists for the function.

> To compute k-ary ψ on inputs $\vec{x}$ do blah blah blah.
>
> By the Church-Turing thesis (CT), there exists an n such that $\psi = \phi_n^k$.

Yippee! But *don't do it until I say you may. And then make sure you provide a procedure.* CT *doesn't read your mind and then write the program you want!*

6.2 Acceptable Indexings[2]

In physics, it is a disaster to confuse "coordinate effects" with physical realities. Thus, the "coriolus force" which "pulls" a projectile westward when it it shot to the north is not a force at all, but the effect of viewing rectilinear inertial motion in a rotating coordinate system. Here is a nice metaphor: programming languages are to computable reality as coordinate systems are to physical reality. The analogy isn't too bad. Physical coordinates allow us to refer to physical events. Programs allow us to refer to computable functions. The arbitrariness of coordinates is handled by the fact that physical laws are invariant under the relevant group of coordinate transformations. What is the corresponding part of the analogy for computable functions? Not physical transformations. Not topological transformations. Not geometrical transformations. You guessed it: computable transformations.

[1]Kleene 1981, op. cit. pp. 59-61.
[2](Rogers, exercise 2-10).

More abstractly, sufficiently powerful computer languages may be viewed as numberings of the partial recursive functions. To be really careful about it, a numbering of $Part$ is a surjective (onto) mapping:

$$\psi_{-}^{-} : \mathbf{N}^2 \to Part$$

where ψ_i^k denotes the i^{th} k-ary partial recursive function according to numbering ψ. Now let δ, ψ be numberings. Say that δ_{-}^{-} compiles into ψ_{-}^{-} just in case for each k there exists a total recursive c_k such that for each i, k, $\delta_i^k = \psi_{c_k(i)}^k$. "Compiles into" is a pre-order (reflexive and transitive). Say that δ_{-}^{-} intercompiles with ψ_{-}^{-} just in case each numbering compiles into the other. Intercompilation is an equivalence relation.

Exercise 6.1 *To make sure you are awake and understand the definitions: prove that compilation is a pre-order (reflexive and transitive) and that intercompilation is an equivalence relation (reflexive, transitive, and symmetric).*

Our special numbering ϕ is not arbitrary. It has a special structure. For example, it satisfies the universal and *s-m-n* theorems. Presumably, it satisfies many other conditions as well. But perversely enough, we will focus on these two curious properties. Say that ψ is acceptable just in case satisfies the conditions of the universal and *s-m-n* theorems; i.e., just in case:

1. $\exists u\ \forall n, \vec{x}\ \psi_u^2(i, \langle \vec{x} \rangle) \simeq \psi_n^{lh(\vec{x})}(\vec{x})$
   (universal machine property);

2. $\forall n, m\ \exists$ total recursive $s\ \forall i, n$-ary $\vec{x}, m$-ary$\vec{y}\ (\psi_{s(i,\vec{x})}^n(\vec{y}) \simeq \psi_i^{m+n}(\vec{x}, \vec{y}))$.
   (*s-m-n* property)

Now why would acceptability be of any interest? Because it characterizes the set of all numberings intercompilable with our original numbering ψ. When you stop to think that the empirical evidence for the Church-Turing thesis is that all natural programming systems yield numberings intercompilable with ψ, we see that the universal and *s-m-n* theorems as it were axiomatize all the computationally invariant structure of our indexing! That means we may kick free of all the arbitrary scaffolding and work only with the universal and *s-m-n* theorems! But first, as they always say, we have to prove it. The proof is a beautiful illustration of how the universal and *s-m-n* constructions interact.

Proposition 6.1 *Numbering* ψ_{-}^{-} *is acceptable* $\iff$ ψ_{-}^{-} *is intercompilable with numbering* ϕ_{-}^{-}.

We proceed by a series of lemmas.

Lemma 6.2 ψ_{-}^{-} *satisfies the universal machine property*
$\Rightarrow \psi_{-}^{-}$ *compiles into* ϕ_{-}^{-}.

Proof. Suppose that $\psi_{\_}^{\_}$ satisfies the universal property. Then for some u,

$$\forall n, \vec{x}\ (\psi_u^2(i, \langle \vec{x} \rangle) \simeq \psi_i^{lh(\vec{x})}(\vec{x})). \tag{6.1}$$

To get rid of the coding on the $\vec{x}$, define:

$$\delta(i, \vec{x}) \simeq \psi_u^2(i, \langle p_m^1(\vec{x}), \ldots, p_m^m(\vec{x}) \rangle). \tag{6.2}$$

Since $rng(\psi_{\_}^{\_}) = Part = rng(\phi_{\_}^{\_})$, there exists a z such that

$$\delta = \phi^m + 1_z. \tag{6.3}$$

Since $\phi_{\_}^{\_}$ has the *s-m-n* property, there is a total recursive s such that for all $i, x, \vec{y}$,

$$\phi_{s(i,x)}^m(\vec{y}) \simeq \phi_i^{m+1}(x, \vec{y}). \tag{6.4}$$

By composing in the constant function c_z, we obtain the total recursive r such that:

$$r(x) = s(z, x). \tag{6.5}$$

Now we use the above to calculate:

$$\begin{array}{rcl}
\phi_{r(i)}^m(\vec{y}) & \simeq & [6.5]; \\
\phi_{s(z,\langle i \rangle)}^m(\vec{y}) & \simeq & [6.4]; \\
\phi_z^{1+m}(i, \vec{y}) & \simeq & [6.3]; \\
\delta(i, \vec{y}) & \simeq & [6.2]; \\
\psi_u^2(i, \langle p_m^i(\vec{x}), \ldots, p_m^m(\vec{x}) \rangle) & \simeq & \psi_u^2(i, \langle \vec{y} \rangle) \\
\psi_u^2(i, \langle \vec{y} \rangle) & \simeq & [6.1] \\
 & \simeq & \psi_i^{lh(\vec{x})}(\vec{x}).
\end{array}$$

Thus, r is the desired, total recursive compiler for arity m. $\dashv$

Lemma 6.3 *$\psi_{\_}^{\_}$ compiles into $\phi_{\_}^{\_}$ $\Rightarrow$ $\psi_{\_}^{\_}$ satisfies the universal property.*

Proof. Suppose that $\psi_{\_}^{\_}$ compiles into $\phi_{\_}^{\_}$. Then for each k there exists a total recursive c such that

$$\psi_i^k \simeq \phi_{c(i)}^k.$$

By the universal theorem for $\phi_{\_}^{\_}$, there is a u such that for all $\vec{y}$:

$$\phi_{c(i)}^m(\vec{y}) \simeq \phi_u^2(c(i), \langle \vec{y} \rangle).$$

So we obtain a partial recursive function

$$\delta(i, \vec{y}) \simeq \phi_u^2(c(i), \langle \vec{y} \rangle).$$

Since $\psi_{\_}^{\_}$ is onto $Part$, there exists a z such that for all $\vec{y}$:

$$\psi_z^2(i, \langle \vec{y} \rangle) \simeq \delta(i, \vec{y}).$$

Thus, for all $\vec{y}$, we have:

$$\begin{array}{rcl}\psi_z^2(i, \langle\vec{y}\rangle) & \simeq & \delta(i, \vec{y}) \\ & \simeq & \phi_u^2(c(i), \langle\vec{y}\rangle) \\ & \simeq & \phi_{c(i)}^m(\vec{y}) \\ & \simeq & \psi_i^k(\vec{y}).\end{array}$$

So z is a universal index for ψ_{-}^{-}. $\dashv$

Lemma 6.4 *ψ_{-}^{-} satisfies the* s-m-n *property* $\Rightarrow$ *ϕ_{-}^{-} compiles into ψ_{-}^{-}.*

Exercise 6.2 *Prove it.*

Lemma 6.5 *ψ_{-}^{-} intercompiles with ϕ_{-}^{-}* $\Rightarrow$ *ψ_{-}^{-} satisfies the* s-m-n *property.*

Exercise 6.3 *Prove it.*