UNDECIDABILITY

Some sets—including many functions and relations—are characterized by a rule or operation that is an **effective method**, i.e., an **algorithm**. Roughly, this means that there is a finite series of well-defined, computer-implementable (or "mechanical") instructions, which can be completed in a finite time, for identifying members of the set.

Where A is any set:

- A is **semi-decidable** or **recursively enumerable** or **effectively enumerable** iff: some algorithm correctly identifies that $x \in A$, for any $x \in A$.
- A is **decidable** or **recursive** or **computable** iff: some algorithm correctly identifies that $x \in A$, for any $x \in A$, *and* it correctly identifies that $y \notin A$, for any $y \notin A$.

  Take heed: Any decidable set also counts as "semi-decidable."

Church's Thesis (CT) says that any intuitively computable set is computable by a Turing Machine (equivalently, by a recursive function; more on these later). CT is not believed to admit of strict mathematical proof, but it is useful in the attempt to make more precise our intuitive notions of "algorithm," "decidable," "computable," etc.

Despite the roughness in these notions, we can prove several non-trivial (and sometimes surprising) results about algorithms, decidability, etc.

51.2: The Algorithms are Countable

Premises:
51.1: English (or any other language) can unambiguously express the series of instructions that define any algorithm, at least if it is supplemented (where necessary) by a finite set of meaningful symbols from other languages.
14.1: There are only countably many finite strings of symbols from a finitary alphabet.

Why believe 51.1? An inexpressible instruction just boggles the mind. (Hunter.)

Proof of 14.1: Let each symbol from your finite alphabet be assigned its own numeric code. Then, any finite string can be coded by the numeral that results from concatenating, in order, the codes for its alphabetic symbols. All finite strings are thereby mapped to a subset of natural numbers, and any subset of natural numbers is countable.

Proof of 51.2: Observe that (supplemented/) English will have a finite alphabet. So by 14.1, the language has only countably many finite strings. Now since any algorithm is defined by a *finitary* series of instructions, then by 51.1, such instructions are always unambiguously expressed by some finite string. Hence, there must be only countably many algorithms.

Corollaries:
51.2 (and 51.3): Some sets of natural numbers are not semi-decidable, hence, undecidable.
51.6: There are only denumerably many computable functions from $\mathbb{N}$ into $\mathbb{N}$.

51.8: The Diagonal Function is Non-Computable

Proof: By 56.1, the set of computable functions from $\mathbb{N}$ into $\mathbb{N}$ is denumerable. Let $\langle f_0, f_1, f_2 \ldots \rangle$ be an enumeration of the computable functions *of one argument* from $\mathbb{N}$ into $\mathbb{N}$. Define the diagonal function g as the total function from $\mathbb{N}$ into $\mathbb{N}$ defined by the rule:

$$g(n) = \begin{cases} 1 & \text{if } f_n(n) = 0 \\ 0 & \text{otherwise.} \end{cases}$$

Suppose for *reductio* that g is computable. Then for some k, $g = f_k$. That is, g would be one of the computable functions in the enumeration $\langle f_0, f_1, f_2 \ldots \rangle$. Now by definition of g, $g(k) = 1$ iff $f_k(k) = 0$. But this means $g(k) = 1$ iff $g(k) = 0$, which is impossible. So g is not a computable function.

Corollary:
51.9: The set $\langle f_0, f_1, f_2 \ldots \rangle$ is not effectively enumerable.

Proof: Suppose otherwise for *reductio*. Then, there would be an algorithm that decides what the jth function is, i.e., what f_j is in the enumeration, for arbitrary j. Further, since all functions in the enumeration are computable, one could then have an algorithm for deciding the value of $f_j(k)$, for any j and k. So in particular, the algorithm would decide the value of $f_n(n)$, for any n. But such an algorithm would suffice to make computable the diagonal function g (defined above). Yet g is not computable. So by *reductio*, $\langle f_0, f_1, f_2 \ldots \rangle$ is not effectively enumerable.

51.10 Respectable Arithmetic is Undecidable

51.10: Any "respectable" formal system of arithmetic is undecidable with respect to its theorems. ["Generalized Undecidability" Theorem, or GU]

> -Peano Arithmetic, Robinson Arithmetic, and Hunter's System H are each "respectable," though it takes much work to show this. Also, since System H is a finite extension of $QS^=$, this leads to the undecidability of $QS^=$ as well. (Church's Theorem.)

Definition. A **formal system of arithmetic** is a formal system such that:

1. Some theorems express truths of number theory, on its intended interpretation.
2. It contains a numeral for each natural number n, written as '$\underline{n}$'.
3. Wff are finite, and composed from a finite alphabet.

Definition: A formal system of arithmetic is **respectable** iff:[1]

(a) The system is consistent.
(b) Every decidable set of natural numbers is *represented* in it (see below).
(c) An open wff is a theorem iff some closure of it is.

Definition. A set X of natural numbers is **represented** in a formal system S iff there is a formula A, with just one free variable v, such that for each natural number *n*:

$\vdash_s A\underline{n}/v$ iff $n \in X$

Proof of GU

Lemma 1: *If S is a formal system of arithmetic, there is a diagonalizing function* g *whose range is not represented in S.*

Where S is a formal system of arithmetic, each one-place formula A of S represents the range of some function *f*, defined as follows:

$$f(n) = \begin{cases} 0 & \text{if } \vdash_s A\underline{n}/v \\ 1 & \text{otherwise.} \end{cases}$$

Since the alphabet is finite, we know (from the coding process in 14.1) that the one-place formulas of S have an effective enumeration $\langle A_0, A_1, A_2 \ldots \rangle$. Let $\langle f_0, f_1, f_2 \ldots \rangle$ be the corresponding enumeration of the functions whose ranges are represented by these formulae. Next, define the diagonalizing function *g* as follows:

$$g(n) = \begin{cases} 1 & \text{if } f_n(n) = 0 \\ 0 & \text{otherwise.} \end{cases}$$

Then, *g* is not in the enumeration $\langle f_0, f_1, f_2 \ldots \rangle$. After all, if *g* were the *k*th function in the enumeration, then $g(k) = f_k(k) = 1$ iff $f_k(k) = 0$, which is impossible.

Lemma 2: *If S is decidable, then G (the range of* g*) is decidable.*
Suppose for conditional proof that the theorems of S are decidable. Then, one could decide whether $\vdash_s A\underline{n}/v$, for any wff $A\underline{n}/v$. That would suffice to make each of $\langle f_0, f_1, f_2 \ldots \rangle$ computable. One could then have an algorithm for deciding the value of $f_j(k)$, for any *j* and *k*. So in particular, the algorithm would decide the value of $f_n(n)$, for any *n.* But such an algorithm would suffice to make computable the diagonalizing function *g.* So its range, G, would be decidable.

The Lemmas Imply: If S is decidable, then G is a decidable set that is *not* represented in S—meaning S is not respectable. So by contraposition, *if S is respectable, then S is undecidable*.

[1] Hunter actually uses 'if' instead of 'iff', so that his definition of 'respectable' is only partial. But we need 'iff' here so to secure the reasoning under "The Lemmas Imply." 'Respectable' seems to be Hunter's own term, and I'm not sure why he hedged it. He may want to avoid controversy in suggesting that any formal arithmetical system where (c) fails is "not respectable." (Lots of legit systems don't have open wff as theorems.) On the other hand, I'm unsure why (c) is needed in the first place. Hunter exploits it in his proof of CD, but the proof below of CD does not.

51.13 Generalized Gödel Theorem

51.13: If S is a respectable formal system of arithmetic with a decidable set of wff and a decidable set of proofs, then S is incomplete, i.e. not negation-complete. [GG]

Premise:
51.12 If S has a decidable set of wff and a decidable set of proofs, then S is consistent and negation-complete only if S is decidable. ["Conditional Decidability" Theorem, or CD]

-GG follows directly from GU and CD.

Proof of CD
Suppose for conditional proof that a consistent, negation-complete system S has a decidable set of wff, and a decidable set of proofs. We can then use Gödel numbering to show that there is an effective enumeration of the proofs $\langle P_0, P_1, P_2 \ldots \rangle$. (Trust me on this for now.) Now given an arbitrary wff A, the negation-completeness of S means that either A or ~A will be a theorem. So run through each of the proofs in the enumeration until you get to one where the last line is A or the last line is ~A. If you find a last line with A then A is a theorem, and (by the consistency of S), ~A is not a theorem. Or if you find a last line with ~A then ~A is a theorem, and (by the consistency of S), A is not a theorem. So, by this decision procedure, S is decidable.