1 Godel numbering

Let $\{a_i\}_{i=0}^n$ a sequence of integers. Let

$$a := \sum_{i=1}^{k} p_i^{a_i+1}$$

We see that $\{a_i\}$ provides a pimre encoding of a, insofar as a can be retrieved from the sequence uniquely (fundametal theorem of arithmetic). This coding is injective but not surjective on ω.

2 Partial function enumeration

To characterize all effectively calculable functions, we would like to produce a list $\mathscr{C} = \{f_n\}_{n\in\omega}$ such that:

- The list includes all and only algorithmically computable functions
- There is an efective listing of said list, i.e. an algorithmically computable function $g(n,x) = f_n(x)$.
- Every f_n is total.

Theorem 1 *Total partial functions are not enumerable.*

Proof: Assume $\mathscr{C}$ satisfies the conditions above. Let $h(x) = g(x,x) + 1$. Since $h = \lambda x \left[Suc \circ \left[g \circ \left[p_1^1, p_1^1\right], p_1^1\right]\right]$, we know h is a computable function. However, we imposed the condition that $g(n,x) = f_n(x)$, which entails $h(x) = f_x(x) + 1 \neq f_x(x)$.

3 Turing

If $M = (Q, \Gamma, \mathscr{b}, \Sigma, \delta, q_1, q_0)$ is a Turing machine, then the map δ is called a *Turing program*. We simplify and assume $\Gamma = \Sigma = \{\mathscr{b}, 1\}$, so that we may refer to a Turing machine as $M = (Q, \delta)$, fixing q_1, q_0 as the start and halting states.

If y is the number of 1s on the tape when M reachs q_0 with input x, and if $\varphi : \omega \to \omega$ is a number-theoretic function s.t. $\varphi(x) = y$, we say M computes φ.

Definition 1 *A Turing computation according to a Turing program P with input x is a sequence of instantaneous description $c_0, \ldots, c_n$ such that c_0 represents the machine in the starting state q_1 reading the leftmost symbol of x, c_n represents the machine in the halting state q_0, and the transition $c_i \to c_{i+1}$ is given by the Turing program P.*

Thus, a *computation* will always refer to a *halting* or *convergent* calculation.

A partial function $\varphi(\overrightarrow{x})$ is represented by letting the input $\overrightarrow{x}$ be the initial configuration:

$$q_1\alpha_1\mathscr{b}\alpha_2 \ldots \mathscr{b}\alpha_n$$

where α_i consists of $x_i + 1$ repetitions of 1.

4 Basic results

4.1 Numbering Turing programs

Let $\{P_e\}_{e\in\omega}$ be an enumeration of all Turing programs, and assuming that P_e is such that e is a Gödel numbering.

Let $\varphi_e^{(n)}$ be a partial function of n variables computed by P_e, and let φ_e denote $\varphi_e^{(1)}$. We from now on identify each turing program P_e with φ_e.

Theorem 2 *Each partial computable function φ_x has infinitely many indices. Furthermore, for each x we can find a set A_x of indices such that $\varphi_y = \varphi_x$ for all $y \in A_x$.*

Proof: Given φ_x, consider the Turing machine P_x. If Turing program P_x mentions only internal states $\{q_0, \ldots, q_n\}$, add any amount of extraneous instructions of the form $q_{n+k}\flat q_{n+k}\flat R$ to create a new program for the same function.

5 Numbering Turing computations

Let

$$c = \overline{t} q_i s_1 \overline{s}$$

denote a Turing computation with $\overline{t} = t_w \ldots t_2 t_1$, s_1 the symbol being read, $\overline{s} = s_2 \ldots s_v$. Thus, the initial configuration is

$$c_1 = q_1 s_1 s_2 \ldots s_{k+1} = q_1 s_1 \overline{s}$$

A Turing machine performs the transitions $c_1, \ldots, c_n$ in deterministic fashion. For every c_i, there is at most one consequent configuratoin c_{i+1}. This passing through an effective sequence of configurations allows for a coding with Gödel numbers as follows.

Once we have assigned numbers to each s_i, q_j, we can assign a number to every configuration using the prime power equation. In short, if $\mathcal{G}(\alpha)$, with $\alpha = q_j$ or $\alpha = s_j$, denotes the numbering of a tape symbol or a state, we let

$$\mathcal{G}(t_m \ldots t_1 q_i s_1 \ldots s_k) = p_1^{\mathcal{G}(q_i)+1} + \sum_{i=1}^{k} p_{i+1}^{\mathcal{G}(s_i)+1}$$

6 Enumeration theorem and universal machines

We know there is no enumeration of all total computable functions. However, the effective numbering for the syntax of Turing machines allows us to give an effective enumeration of all partial computable functions.

Theorem 3 *There is a partial computable function ψ of two arguments such that $\psi(e, x) = \varphi_e(x)$. Turing's thesis ensures there is some i such that $\varphi_i(e, x) = \psi(e, x)$.*

Proof: Let $M(e,x)$ be the Turing machine which computes $\psi(e,x)$ and call it universal Turing machine. We provide an effective procedure which can, by TT, be translated into a Turing program.

> (1) Given input (e,x), convert e to P_e by using the numbering of Turing programs.
>
> (2) Simulate P_e on input x. The simulation begins with state q_1 on the leftmost symbol of x in standard input form. If the simulation is in state q_j reading symbol s_k, then M searches through the tuples (instructions) in P_e until it finds the indicated action on the simulation tape.

The enumeration theorem is crucial insofar as it guarantees the existence of a universal computational machine, capable of running any computable program.

7 Parameter theorem

Theorem 4 (Parameter theorem) *There is a bijective computable function $\mathfrak{z}(x,y)$ such that, for all $x,y,z \in \omega$,*

$$\varphi_{\mathfrak{z}(x,y)}(z) = \varphi_x(y,z)$$

The theorem specifies that, given the Godel numbering x of a function φ_x of two parameters, we can fix the first parameter y and computably find $\mathfrak{z}(x,y)$ satisfying that

$$\varphi_{\mathfrak{z}(x,y)}(z) = \varphi_x(y,z)$$

Example 1 *Let $\varphi_x(y,z) = y + z$. Fix $y = 3$ and consider the resulting function $f(z) = \varphi_x(3,z) = 3 + z$. Now $f(x)$ is computable and must have some index i such that $\varphi_i(z) = f(z)$. The parameter theorem shows that we can pass computably from x and the parameter y to such an index i for $f(z)$.*

Proof: Let x be the Godel numbering of a function φ_x of two parameters, and P_x be the Turing program which computes said function. Let $\mathcal{S}_y$ be a Turing machine which takes as input a Turing program P, a number z, and computes P with input (y,z). Then clearly, if y is held constant, $\mathcal{S}_y$ with input P_x and z computes $\varphi_x(y,z)$. Clearly, $\mathcal{S}_y$ corresponds to a partial computable function with one parameter, and it depends directly on x and y; i.e. $\mathcal{S}_y$ corresponds to a partial function $\varphi_{\mathfrak{z}(x,y)}(z)$.

Alternative proof: An alternative proof comes from von Neumann's paradigm. Let $\mathbb{Q}$ be a program which computes $f(y,z)$. Let $\mathcal{P}_y$ be the program which, on input z, sets $N1 \leftarrow y$, treating y as a constant, and $N2 \leftarrow z$. Then, evidently, the concatenation $\mathcal{P}_y\mathbb{Q}$ computes $f(y,z)$. Which entails

$$\Psi_{\mathbb{Q}}(y,z) \simeq \Psi_{\mathcal{P}_y\mathbb{Q}}(z) \blacksquare$$

Theorem 5 (Unbounded search) *If $\theta(x,y)$ is a partial computable function, and*

$$\psi(x) = (\mu y)\,[\theta(x,y)\downarrow = 1 \wedge (\forall z < y)\,[\theta(x,z)\downarrow \neq 1]]$$

then $\psi(x) = y$ is a partial computable function.

Proof: For a fixed x_0, compute $\theta(x_0, y)$ as follos. For fixed s, compute s steps for each $y \leq s$ and then proceed to step $s+1$. Continue until (if ever) the first y is found such that $\theta(x, y) \downarrow = 1$. Output $\psi(x) = y$.

7.1 Unsolvable problems

Convention. If $R \subseteq \omega^n$, then R has the property P if the set $\{\langle x_1, \ldots, x_n\rangle : R(x_1, \ldots, x_n)\}$ has property P, such as being computable.

Definition 2 *A set is computably enumerable if it is the domain of a partial computable function.*

Let

$$W_e := \mathcal{D}_{\varphi_e} = \{x : \varphi_e(x) \downarrow\}$$

Any computable set A is computably enumerable, since if A is computable then $A = \mathcal{D}(\chi_A)$. A theorem establishes that a non-empty set is computably enumerable iff it is the range of a computable function, i.e. if there is an algorithm for listing its members.

Though all computable sets are computably enumerable, not all computably enumerable sets are computable.

Definition 3 *Let* $K := \{x : \varphi_x(x)$ *is defined*$\} = \{x : x \in W_x\}$.

Theorem 6 *K is computably enumerable.*

Proof: The enumeration theorem guarantees that $\varphi_x(x) = \psi(x, x)$ for some partial computable function ψ, and K is the domain $\theta(x) = \psi(x, x)$.

Theorem 7 *K is not computable.*

Proof:

W_x is not computable. If W_x had a computable characteristic function, the following function would be computable:

$$f(x) = \begin{cases} \varphi_x(x) + 1 & x \in W_x \\ 0 & x \notin W_x \end{cases}$$

However, f cannot be computable because $f \neq \varphi_x$ for every x.

Definition 4 *We define* $K_0 := \left\{\langle x, y\rangle : x \in W_y\right\}$.

Theorem 8 *K_0 is computably enumerable.*

Proof: K_0 is a set of encodings of (x, y) pairs s.t. x is in the domain of φ_y. We know $\varphi_y(x) = \psi(y, x)$ due to the enumeration theorem, where ψ is computable. Defining $\theta(\langle x, y\rangle) = \psi(y, x)$, we ensure that $\theta(\langle x, y\rangle)$ is defined iff $\psi(y, x)$ is defined, which happens iff $x \in \mathcal{D}_{\varphi_y}$.

$\therefore K_0 = \mathcal{D}_\theta$.

Theorem 9 *K_0 is not computable.*

Proof: Note that $x \in K$ iff $\langle x, x\rangle \in K_0$. Thus, if K_0 had a computable characteristic function, so would K.

The halting problem consists precisely of deciding, for arbitrary x and y, whether $\varphi_y(x)$ converges, i.e. whether $\langle x, y\rangle \in K_0$. The previous theorem ensures the unsolvability of the halting problem.

Definition 5 *We write $A \leq_m B$ to say there is a computable f such that $f(A) \subseteq B$ and $f(\overline{A}) = \overline{B}$. In other words, if $x \in A$ iff $f(x) \in B$.*

If said f is a bijection, we write $A \leq_1 B$.

Definition 6

- *We say $A \equiv_m B$ if $A \leq_m B$ and $B \leq_m A$, and equivalently for $A \equiv_1 B$.*
- *$deg_m(A) = \{B : A \equiv_m B\}$, and equivalently for $deg_1(A)$.*

Observe that $\equiv_m, \equiv_1$ define an equivalence relation. Their equivalence classes are called the m-degrees and 1-degrees, respectively.

Theorem 10 *If $A \leq_m B$ and B is computable, then A is computable.*

Proof: χ_B denotes the belonging to B. However, for any $a \in A$, we know $a \in A$ iff $f(a) \in B$. So $\chi_A(x) = \chi_B(f(x))$.

If an unsolvable problem A can be reduced to another problem B, then B is also unsolvable.

Theorem 11 *$K \leq_1 \{x : \varphi_x$ is total$\}$*

Proof: Let $\mathcal{T} := \{x : \varphi_x$ is total$\}$. Define

$$\psi(x, y) = \begin{cases} 1 & x \in K \\ \text{undefined} & \text{otherwise} \end{cases}$$

Clearly, ψ is partially computable because the program to compute $\psi(x, y)$ says: attempt to compute $\varphi_x(x)$: if this fails to converge, output nothing; if it converges, then output 1 for every argument y.

The parameter theorem ensures there is a bijective computable function f such that

$$\psi(x, y) = \varphi_{f(x)}(y)$$

Choose e such that $\varphi_e(x, y) = \psi(x, y)$, and define

$$f(x) = \lambda x \left[s_1^1(e, x)\right]$$

Clearly, f s one to one. Note that

- $x \in K \rightarrow \varphi_{f(x)} = \lambda y\,[1] \Rightarrow \varphi_{f(x)}$ total $\Rightarrow f(x) \in \mathcal{T}$.