0.0.1 Proposition. *$\mathcal{P}$ is closed under unbounded search; that is, if $\lambda x\vec{y}.g(x,\vec{y})$ is in $\mathcal{P}$, then so is $\lambda\vec{y}.(\mu x)g(x,\vec{y})$.*

Proof. See Notes #2. □

Why "unbounded" search? Because we do not know *a priori* how many times we have to go around the loop. This depends on the behavior of g.

Before we get more immersed into *partial functions* let us **redefine equality for function calls**.

0.0.2 Definition. Let $\lambda\vec{x}.f(\vec{x}_n)$ and $\lambda\vec{y}.g(\vec{y}_m)$.

We extend the notion of equality $f(\vec{a}_n) = g(\vec{b}_m)$ to include the case of *undefined calls*:

For any $\vec{a}_n$ and $\vec{b}_m$, $f(\vec{a}_n) = g(\vec{b}_m)$ means precisely one of

- For some $k \in \mathbb{N}$, $f(\vec{a}_n) = k$ and $g(\vec{b}_m) = k$
- $f(\vec{a}_n)\uparrow$ and $g(\vec{b}_m)\uparrow$

For short,

$$f(\vec{a}_n) = g(\vec{b}_m) \equiv (\exists z)\Big(f(\vec{a}_n) = z \wedge g(\vec{b}_m) = z \vee f(\vec{a}_n)\uparrow \wedge g(\vec{b}_m)\uparrow\Big)$$

□

0.0.3 Lemma. *If $f = prim(h,g)$ and h and g are **total**, then so is f.*

The definition is due to Kleene and he preferred, as I do in the text, to use a new symbol for the extended equality, namely $\simeq$.

Regardless, by way of this note we will use the same symbol for equality for **both** total and nontotal calls, namely, "=" (this conventions is common in the literature, e.g., [Rog67]).

Proof. Let f be given by:

$$\begin{aligned} f(0,\vec{y}) &= h(\vec{y}) \\ f(x+1,\vec{y}) &= g(x,\vec{y},f(x,\vec{y})) \end{aligned}$$

We do induction on x to prove

$$\text{"For all } x, \vec{y},\ f(x,\vec{y})\downarrow\text{"} \tag{$*$}$$

Basis. $x = 0$: Well, $f(0,\vec{y}) = h(\vec{y})$, but $h(\vec{y})\downarrow$ for all $\vec{y}$, so

$$f(0,\vec{y})\downarrow \text{ for all } \vec{y} \tag{$**$}$$

As I.H. (Induction *Hypothesis*) take that

$$f(x,\vec{y})\downarrow \text{ for all } \vec{y} \text{ and fixed } x \tag{$\dagger$}$$

Do the Induction *Step* (I.S.) to show

$$f(x+1, \vec{y}) \downarrow \text{ for all } \vec{y} \text{ and the fixed } x \text{ of } (\dagger) \tag{$\ddagger$}$$

Well, by (†) and the assumption on g,

$$g\big(x, \vec{y}, f(x, \vec{y})\big) \downarrow, \text{ for all } \vec{y} \text{ and the fixed } x \text{ of } (\dagger)$$

which says the same thing as (‡). Having proved the latter and the Basis, (∗∗), we have proved (∗) by induction on x. □

0.0.4 Corollary. *$\mathcal{R}$ is closed under primitive recursion.*

Proof. Let h and g be in $\mathcal{R}$. Then they are in $\mathcal{P}$. But then $prim(h, g) \in \mathcal{P}$ as we showed in class/text and Notes #2. By 0.0.3, $prim(h, g)$ is total. By definition of $\mathcal{R}$, as **the subset of $\mathcal{P}$ that contains all total functions of $\mathcal{P}$**, we have $prim(h, g) \in \mathcal{R}$. □

Why all this dance **in colour** above? Because to prove $f \in \mathcal{R}$ you need **TWO** things: That

1. $f \in \mathcal{R}$

   AND

2. f is total

But aren't all the total functions in $\mathcal{R}$ anyway?

NO! They need to be computable too! I.e., of the form $M_{\mathbf{y}}^{\vec{\mathbf{x}}_n}$ for some URM M.

Aren't they all?

NO! See next section, and heed the last sentence in the last -remark!

0.1 Diagonalisation

We start with an example.

0.1.1 Example. Suppose we have a 3×3 matrix

$$\begin{array}{ccc} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{array}$$

and we are asked: Find a sequence of three numbers, *using only* 0 *or* 1, that does not *fit* as a row of the above matrix—i.e., is *different from all rows*.

Sure, you reply: Take 0 0 0.

That is correct. But what if the matrix were big, say, 100×100, or $10^{350000} \times 10^{350000}$, or even *infinite*?

Is there a *finitely describable technique* that can produce an "unfit" row for any square matrix, no mater how big; even for an infinite one?

Yes, it is Cantor's "*diagonal method*" or "diagonalisation" which he introduced in his famous "Set Theory".

He noticed that any row that fits in the matrix as the, say, i-th row, intersects the main diagonal at the same spot that the i-th column does.

Thus if we take the main diagonal —a sequence that has the same length as any row— and *change every one of its entries*, then it will not fit anywhere as a row! Because no row can have an entry that is different than the entry at the location where it intersects the main diagonal!

This idea would give the answer 0 1 0 to our original question.

While the array 1000 11 3 also follows the principle of making every entry on the diagonal different than the original, and works, we were constrained in this example to "using only 0 or 1", else one could also "cheat" and provide "42 42 42" as an example that does not fit, since no entry is 0 or 1.

More seriously, in a case of a very large or infinite matrix it is best to have a *simple technique* that *works* even if we do not know much about the elements of the matrix. Read on! □

0.1.2 Example. We have an infinite matrix of 0-1 entries. Can we produce an *infinite sequence* of 0-1 entries that does not match any row in the matrix?

Pause. What is an *infinite sequence*? Our intuitive understanding of the term is captured **mathematically** by the concept of a **total** function f with left field (and hence domain) $\mathbb{N}$. The n-th member of the sequence is $f(n)$.◀

Yes, take the main diagonal and flip every entry (0 to 1; 1 to 0).

Now, the diagonal entries have matrix coordinates (i, i) for $i = 0, 1, 2, \ldots$

Note that row i of the matrix intersects the diagonal at entry (i, i), in other words,

$$\text{the entry } i \text{ of row } i \text{ is the matrix entry } (i, i) \qquad (*)$$

So, can this constructed 0-1 array —let's call it d— fit as row i, for some i?

If yes, then, by $(*)$, $d(i)$ equals the matrix entry at (i, i) —let's say, a.

But **by construction** of d, $d(i) = 1 - a$. Since $a \neq 1 - a$ we have **NO fit**! Thus d fits nowhere, i being arbitrary.

You can see clearly now why the Cantor technique demonstrated here and in the previous example is called "*the diagonal method*" or "*diagonalisation*". It uses the diagonal of a matrix in a clever and simple way. □

0.1.3 Example. We have an infinite matrix of entries from $\mathbb{N}$ (many may be > 1). Can we produce an infinite sequence of $\mathbb{N}$-entries that does not match any row in the matrix? Yes, take the main diagonal and **change** every entry from a to $a + 1$.

If the original diagonal has an a in row i, the constructed row has an $a + 1$ in column i, so it will not fit as row i since $a \neq a + 1$. So it fits nowhere, i being arbitrary.

Seeing that an infinite numerical array is the (sorted by input) sequence of outputs of a **total** function f with left and right fields equal to $\mathbb{N}$, this example shows that if we have a sequence of such one-argument functions, say

$$f_0, f_1, f_2, f_3, \ldots$$

then these define the infinite matrix

$$\begin{array}{cccccc}
f_0(0) & f_0(1) & f_0(2) & \ldots & f_0(i) & \ldots \\
f_1(0) & f_1(1) & f_1(2) & \ldots & f_1(i) & \ldots \\
\vdots & & & & & \\
f_i(0) & f_i(1) & f_i(2) & \ldots & f_i(i) & \ldots \\
\vdots & & & & &
\end{array}$$

The procedure in blue type above constructs a function $d = \lambda x.1 + f_x(x)$, which as an array,

$$d(0), d(1), \ldots$$

does not fit in the matrix anywhere as a row —because $d(i)$ is different from $f_i(i)$ for all i.

That is, we constructed a function d that <u>cannot be one of the f_i</u>! □

0.1.4 Example. (Cantor's original theorem, somewhat amended) Let S denote the set of **all** infinite sequences of 0s and 1s.

Can we arrange *all* of S in an infinite matrix —one element per row?

No, since the preceding example 0.1.2 shows how to **construct** an infinite 0-1 sequence that is NOT possibly a row of the matrix.

Thus would miss at least one infinite sequence (i.e., we would fail to list it as a row), namely **the one constructed by diagonalisation**.

But arranging all members of S as an infinite matrix—one element per row—is tantamount to saying that we can enumerate all the members of S using members of $\mathbb{N}$ as indices.

So we cannot do that.

In Set Theory jargon we say, *S is **un**countable* or also ***not** enumerable*.

By contrast, a set is countable or enumerable if it can be so enumerated as an infinite sequence. From the definition of "infinite sequence" this means that a set S is countable iff for some total f with domain $\mathbb{N}$ we have $S = ran(f)$.

BTW, the amendment we did to Cantor's theorem here is twofold.

- He did not care about sequences; he wanted to show that the set of reals in the unit interval, $[0, 1] \stackrel{Def}{=} \{x \in \mathbb{R} : 0 \leq x \leq 1\}$, is uncountable. Well, any such real IS essentially an infinite "binary sequence" that starts with a dot "." ("is essentially" means "is represented by")
- Cantor actually used base-10, not base-2 representation of the reals in $[0, 1]$.

$\square$

Example 0.1.4 shows that uncountable sets exist. Here is a more interesting one.

0.1.5 Example. (0.1.2 Retold) Consider the set $\mathcal{T}_{\{0,1\}}$ of **all total** functions from $\mathbb{N}$ to $\{0, 1\}$. Is this countable?

Well, if there is an enumeration of these one-variable functions

$$f_0, f_1, f_2, f_3, \ldots \tag{1}$$

consider the function $g : \mathbb{N} \to \{0, 1\}$ given by $d(x) = 1 - f_x(x)$. Clearly, this *must* appear in the listing (1) since it has the correct left and right fields, and is total.

Too bad! If $d = f_i$ then $d(i) = f_i(i)$, by evaluating both sides at i. However, by definition of d, we also have $d(i) = 1 - f_i(i)$. A contradiction since $f_i(i) \neq 1 - f_i(i)$.

For the contradiction it is crucial that the f_i are total! For if, say, $f_i(i)\uparrow$ then we get no contradiction as $f_i(i) = 1 - f_i(i)$ in this case! (Cf. Definition 0.0.2.

The above argument is a "mathematized" version of 0.1.2; as already noted, an infinite sequence of 0s and 1s is just a total function from $\mathbb{N}$ to $\{0, 1\}$. $\square$

0.1.6 Remark. An analogous argument to the above shows that the set of all **total** functions from $\mathbb{N}$ to $\mathbb{N}$ is also uncountable.

Indeed, taking $d(x) = f_x(x) + 1$ as in 0.1.3 works in this case to "systematically change the diagonal" $f_0(0), f_1(1), \ldots$ since we are not constrained to keep the function values in $\{0, 1\}$. Indeed, **IF** $d = f_i$ —that is, the "array" d fits as row number i— then

$$f_i(i) + 1 \stackrel{\text{construction}}{=} d(i) \stackrel{\text{assumption just stated}}{=} f_i(i)$$

Hence $f_i(i) + 1 = f_i(i)$, a contradiction since both sides are defined. This mathematises the technique in Example 0.1.3. □

0.2 A digression regarding $\mathcal{R}$

Add the symbol ";" (without the quotes) to the URM alphabet. Use ; as inter-instruction glue to turn a URM written vertically —one instruction per line— into a "horizontal" string ofg symbols.

Easy to believe (and verify) fact (with a pseudo program, or indeed one written in C or JAVA) that

> *We can computationally* test *if a string over the augmented alphabet is a syntactically correct URM or not.*

But then we can **enumerate** (e.g., *put in a growing list*), indeed computationally, all URMs as follows:

1. Enumerate *the next* string over the augmented alphabet, choosing "next" in the *lexicographic* order. **Incidentally, this can be done via a program**.

2. *For each string generated* above **do**: *Test* it whether it is a URM or not. If *not*, **Goto** 1.

   If *yes*, then add it to the growing list of URMs and then **Goto** 1.

You can now enumerate all partial recursive functions of one variable!

Simply, for each URM M added to the list, enumerate all $M_{\mathbf{y}}^{\mathbf{x}}$ for **all** pairs of variables $(\mathbf{x}, \mathbf{y})$ found in M. Do so lexicographically (recall that $\mathbf{x}$ is really a string of the form $X11\ldots1$). Examples of $(\mathbf{x}, \mathbf{y})$ pairs: "$(X11, X11)$", "$(X11, X1)$, $(X11111111, X111)$". **This enumeration too can be done algorithmically!**

So, the set of all $\mathcal{P}$ functions of one variable is countable.

The above sentence in blue type says less than what we proved in outline: We proved that the enumeration is computable, we showed!

But then, as $\mathcal{R} \subset \mathcal{P}$,

$$\textbf{the set of all } \mathcal{R} \textbf{ functions of one variable is countable.} \qquad (\dagger)$$

Indeed, in the enumeration of the $\mathcal{P}$ functions of one variable just *omit* the non total ones!

This latter enumeration is just *mathematical*. We will see later that it *cannot* be done computationally, but we do not care for the goal in hand here:

We saw that the set of **all** total functions of one variable is **uncountable** (0.1.6). Thus, $\mathcal{R}$ is a **proper** subset of since a set cannot be both countable and uncountable.

That is, there are total functions that are not URM-computable.

This is why we always warn: When one wants to prove that $f \in \mathcal{R}$ one must do **two** things! One is that f is computable (in $\mathcal{P}$). **Never take this for granted**!

Bibliography

[Rog67] H. Rogers, *Theory of Recursive Functions and Effective Computability*, McGraw-Hill, New York, 1967.