

March 19?

### 0.0.1 Reducibility via the S-m-n Theorem

We now turn to the development of the technique of reductions, using the S-m-n theorem.

**0.0.1 Definition. (Strong Reducibility)** We say that the set  $A$  (subset of  $\mathbb{N}$ ) is *strongly reducible* to set  $B$ , in symbols  $A \leq_m B$ , iff there is a recursive function  $f$  such that for all  $x$ , we have  $x \in A$  iff  $f(x) \in B$ . We say that  $f$  *effects the reducibility*.  $\square$




**0.0.2 Remark.** Several remarks are in order:

- (1) The  $m$  in the reducibility symbol reflects the fact that  $f$  is not required to be 1-1. So, strong reducibility, by default, is a *many-one reducibility* or *m-reducibility*.

We also have *1-1 reducibility* or *1-reducibility*. This is when  $f$  is 1-1.

- (2) The condition  $x \in A$  iff  $f(x) \in B$  says that if we know how to decide  $z \in B$  and also know how to compute  $f(x)$  for all  $x$ , then we know how to decide  $x \in A$ . Thus, the symbol  $\leq_m$  is apt: Intuitively  $A$  is “more solvable” than  $B$  since we can decide it if we can decide  $B$ . Conversely,  $B$  is **more unsolvable than  $A$** .

We express this technically in the proposition below.

- (3) By definition,  $A \leq_m B$  iff  $A = \{x : f(x) \in B\}$ . That is, iff  $A = f_{\leftarrow}(B)$ .  $\square$  



**0.0.3 Proposition.** *Suppose that  $A \leq_m B$ . Then*

- (1)  *$A$  is recursive if  $B$  is. Contrapositively,  $B \notin \mathcal{R}_*$  if  $A \notin \mathcal{R}_*$ .*
- (2)  *$A$  is semi-computable if  $B$  is. Contrapositively,  $B$  is not c.e. if  $A$  is not c.e.*


*Proof.*

- (1) If  $z \in B$  is recursive, then so is  $f(x) \in B$  by the assumption on  $f$  and by the theorem on substitution of function calls into recursive predicates.
- (2) If  $z \in B$  is semi-recursive, then so is  $f(x) \in B$  by the assumption on  $f$  and by the theorem on substitution of function calls into semi-recursive predicates.  $\square$

**0.0.4 Definition. (Complete Index Sets)** Given a subset  $\mathcal{C} \subseteq \mathcal{P}$ , we call  $\{x : \phi_x \in \mathcal{C}\}$  a *complete index set* (defined by  $\mathcal{C}$ ).  $\square$

 **0.0.5 Remark.** That is, a complete index set  $A = \{x : \phi_x \in \mathcal{C}\}$  is the set of *all* (codes of) URM programs that compute the functions of some given subset  $\mathcal{C}$  of  $\mathcal{P}$ . Indeed say  $f \in \mathcal{C}$ . As this is computable, take *any* program  $i$  for  $f$ , that is,  $f = \phi_i$ . Now  $\phi_i \in \mathcal{C}$  yields  $i \in A$  by the definition of  $A$ .  $\square$  

This subsection deals with the undecidability of membership in several complete index sets. Indeed, “several” is an understatement. We will conclude with the rather surprising Theorem of Rice, according to which the *only* decidable such problems involve the *two* trivial cases:  $\mathcal{C} = \emptyset$  or  $\mathcal{C} = \mathcal{P}$ .

 **0.0.6 Remark. (The General Technique)** The technique in general outline goes like this: We want to show that some  $A$  given as  $\{x : \phi_x \in \mathcal{C}\}$ , for some  $\mathcal{C} \subseteq \mathcal{P}$ , is not recursive.

Equipped with 0.0.3 we attempt to show either  $K \leq_m A$  or  $\overline{K} \leq_m A$  — whichever is easier. The latter, of course, yields more information (a stronger result): that  $A$  is not c.e.

To this end, we need to demonstrate that there is an  $h \in \mathcal{R}$  that effects one of  $K \leq_m A$  or  $\overline{K} \leq_m A$ .

To execute this plan, we utilize the S-m-n theorem so that we come up with a primitive recursive  $h$  such that


Case of  $K \leq_m A$ :

$$\phi_{h(x)} = \begin{cases} \text{some specific } f \in \mathcal{C} & \text{if } \phi_x(x) \downarrow \\ \text{some specific } g \notin \mathcal{C} & \text{if } \phi_x(x) \uparrow \end{cases}$$

Thus,  $h(x) \in A$  iff the top case holds, iff  $x \in K$ —that is,  $\phi_x(x) \downarrow$ . For short,  $K \leq_m A$  via  $h$ .

Case of  $\overline{K} \leq_m A$ :

$$\phi_{h(x)} = \begin{cases} \text{some specific } f \in \mathcal{C} & \text{if } \phi_x(x) \uparrow \\ \text{some specific } g \notin \mathcal{C} & \text{if } \phi_x(x) \downarrow \end{cases}$$

Thus,  $h(x) \in A$  iff the top case holds, iff  $x \in \overline{K}$ —that is,  $\phi_x(x) \uparrow$ . For short,  $\overline{K} \leq_m A$  via  $h$ .  $\square$  

The following theorem is important both in content and in regards to **learning the technique employed for its proof**.

**0.0.7 Theorem.** *The following sets are not recursive.*

- (1)  $A = \{x : \phi_x \text{ is a constant function}\}$
- (2)  $B = \{x : \phi_x \text{ is total}\} = \{x : \phi_x \in \mathcal{R}\}$
- (3)  $C = \{(x, y) : y \in \text{ran}(\phi_x)\}$
- (4)  $D = \{(x, y, z) : z = \phi_x(y)\}$
- (5)  $E = \{x : \text{dom}(\phi_x) = \emptyset\}$
- (6)  $F = \{x : \text{dom}(\phi_x) \text{ is finite}\}$
- (7)  $G = \{x : \text{dom}(\phi_x) \text{ is infinite}\}$
- (8)  $H = \{x : \text{ran}(\phi_x) = \emptyset\}$
- (9)  $I = \{x : \text{ran}(\phi_x) \text{ is finite}\}$
- (10)  $J = \{x : \text{ran}(\phi_x) \text{ is infinite}\}$

*Proof.*

- (1)  $A = \{x : \phi_x \text{ is a constant function}\}$ .

We will be more expansive in just this *first* case. Following 0.0.6, we want to find an  $h \in \mathcal{R}$  such that

$$\phi_{h(x)} = \begin{cases} \text{some specific constant function} & \text{if } \phi_x(x) \downarrow \\ \text{some specific non constant function} & \text{if } \phi_x(x) \uparrow \end{cases} \quad (\dagger)$$



Well, the simplest solution is probably this: Define, for all  $x$  and  $y$

$$\psi(x, y) = \begin{cases} 0 & \text{if } \phi_x(x) \downarrow \\ \uparrow & \text{if } \phi_x(x) \uparrow \end{cases} \quad (*)$$

We see first at the intuitive level that  $\psi$  is computable: Given  $x, y$ . We ignore  $y$ . Next, we fetch the URM  $M$  of code  $x$  and call it on input  $x$ . If it ever halts, then we print “0” and halt everything. If  $M$  never halts, then our process will never return from the call, which is the correct behavior for  $\psi(x, y)$ —bottom case.

Intuitively, we cannot expect to yield some output in the bottom case, since, at least in the process described above, the call to  $M$  for input  $x$  will never halt to give us an opportunity to print anything.

**Pause.** Do you have a reason as to why defining  $\psi$  so that, say, it — *mathematically* speaking— yields 42 in the bottom case renders  $\psi$  non computable (i.e., not in  $\mathcal{P}$ )? ◀

  $\psi \in \mathcal{P}$  via definition by positive cases. The last (bottom) case is the “otherwise” case. 

By the S-m-n theorem, there is an  $h \in \mathcal{PR}$  such that, for all  $x$  and  $y$ ,

$$\phi_{h(x)}(y) = \begin{cases} 0 & \text{if } \phi_x(x) \downarrow \\ \uparrow & \text{if } \phi_x(x) \uparrow \end{cases} \quad (**)$$

This can be rewritten as

$$\phi_{h(x)} = \begin{cases} \lambda y.0 & \text{if } \phi_x(x) \downarrow \\ \emptyset & \text{if } \phi_x(x) \uparrow \end{cases} \quad (***)$$



where  $\emptyset$  is the empty function —clearly not a constant function! We have achieved the setup (†) and we conclude by directly invoking 0.0.6.

(2)  $B = \{x : \phi_x \text{ is total}\} = \{x : \phi_x \in \mathcal{R}\}.$

Note that (\*\*\*) can be recast as

$$\phi_{h(x)} = \begin{cases} \text{a specific total } f & \text{if } \phi_x(x) \downarrow \\ \text{a specific nontotal } g & \text{if } \phi_x(x) \uparrow \end{cases}$$

Thus,  $K \leq_m B$  and therefore  $B \notin \mathcal{R}_*$  as in 0.0.6.

 This result says less than what we already proved earlier, that is,  $B$  is not c.e., however, it is important to see this alternative technique even if (seemingly) achieves less. **Seemingly.** We will refine the technique in the next theorem, to redidcover the non semi-recursiveness of  $B$ . 

(3)  $C = \{(x, y) : y \in \text{ran}(\phi_x)\}.$

Let us use the present technique. If  $C$ , i.e.,  $y \in \text{ran}(\phi_x)$ , is recursive, then so is  $0 \in \text{ran}(\phi_{h(x)})$  by Grzegorzczk Ops, where  $h$  is the same as above. Let us set  $C_0 = \{x : 0 \in \text{ran}(\phi_{h(x)})\}.$

But  $x \in C_0$  is true iff we are in the top case of (\*\*\*), which is the case  $x \in K$ .

**Thus  $K = C_0$ ; not in  $\mathcal{R}_*$  as we know.**

(4)  $D = \{(x, y, z) : z = \phi_x(y)\}.$

If  $D$  is recursive, then so is  $D_0 = \{x : 0 = \phi_{h(x)}(0)\}.$  **The predicate  $0 = \phi_{h(x)}(0)$  is equivalent to  $x \in K$  as above, so it is not in  $\mathcal{R}_*$ .**

(5)  $E = \{x : \text{dom}(\phi_x) = \emptyset\}.$

We can still mine diverse unsolvability results from the very same setup (\*\*\*) above. We rewrite this as

$$\phi_{h(x)} = \begin{cases} \text{a } g \text{ with a non-empty domain} & \text{if } \phi_x(x) \downarrow \\ \text{an } f \text{ with an empty domain} & \text{if } \phi_x(x) \uparrow \end{cases}$$

Thus, as in 0.0.6,  $h(x) \in E$  iff we are in the **bottom case**; iff  $x \in \overline{K}$ . That is,  $\overline{K} \leq_m E$  via  $h$ . We have proved more than what we were asked to:  $E$  is not even semi-recursive, let alone decidable.

- (6)  $F = \{x : \text{dom}(\phi_x) \text{ is finite}\}$ .

We rewrite (\*\*\*) as

$$\phi_{h(x)} = \begin{cases} \text{a } g \text{ with an infinite domain} & \text{if } \phi_x(x) \downarrow \\ \text{an } f \text{ with a finite domain} & \text{if } \phi_x(x) \uparrow \end{cases}$$

Thus,  $h(x) \in F$  iff we are in the bottom case; iff  $x \in \overline{K}$ . That is  $\overline{K} \leq_m F$  via  $h$ . Once again we have proved more than we were asked to:  $F$  is not semi-recursive.

- (7)  $G = \{x : \text{dom}(\phi_x) \text{ is infinite}\}$ .

Yet again, we rely on (\*\*\*). Indeed, see the argument for  $F$  above. We have that  $h(x) \in G$  iff we are in the *top* case; iff  $x \in K$ . That is,  $K \leq_m G$  via  $h$  and thus  $G$  is not recursive.

- (8)  $H = \{x : \text{ran}(\phi_x) = \emptyset\}$ .

One last time we mine (\*\*\*), rewriting it as

$$\phi_{h(x)} = \begin{cases} \text{a } g \text{ with a non-empty range} & \text{if } \phi_x(x) \downarrow \\ \text{an } f \text{ with an empty range} & \text{if } \phi_x(x) \uparrow \end{cases}$$

Thus,  $h(x) \in H$  iff we are in the bottom case; iff  $x \in \overline{K}$ . Thus  $\overline{K} \leq_m H$  and  $H$  is not semi-recursive.

- (9)  $I = \{x : \text{ran}(\phi_x) \text{ is finite}\}$ .

This case needs a fresh start, since neither  $\lambda y.0$  nor  $\emptyset$  have an infinite *range*, as needed for the “dichotomy” *infinite* vs. *finite* (range), toward applying the technique of 0.0.6. So, we define a new function, for all  $x$  and  $y$ , by

$$\chi(x, y) = \begin{cases} y & \text{if } \phi_x(x) \downarrow \\ \uparrow & \text{otherwise} \end{cases}$$

$\chi$  is computable. Intuitively, given  $x$  and  $y$  we decode  $x$  to get the URM  $M$  that it codes. We then call  $M$  on input  $x$ . If it ever halts, we print  $y$  and halt all; otherwise we keep going.

Mathematically,  $\chi$  is defined by positive cases, since  $\phi_x(x) \downarrow$  is c.e. Thus it is in  $\mathcal{P}$ . The S-m-n theorem guarantees the existence of a  $k \in \mathcal{R}$ , such that, for all  $x$  and  $y$ ,

$$\phi_{k(x)}(y) = \begin{cases} y & \text{if } \phi_x(x) \downarrow \\ \uparrow & \text{otherwise} \end{cases}$$

---

Put more conveniently, with no reference to inputs,

$$\phi_{k(x)} = \begin{cases} \lambda y.y & \text{if } \phi_x(x) \downarrow \\ \emptyset & \text{otherwise} \end{cases} \quad (\ddagger)$$

Note that  $k(x) \in I$  iff we are in the bottom case of  $(\ddagger)$ ; iff  $x \in \overline{K}$ . Thus  $\overline{K} \leq_m I$  via  $k$ , rendering  $I$  non c.e., which says more than what we set out to prove.

(10)  $J = \{x : \text{ran}(\phi_x) \text{ is infinite}\}$ .

We reuse  $(\ddagger)$ . Here  $k(x) \in J$  iff we are in the *top* case of  $(\ddagger)$ ; iff  $x \in K$ . Thus  $K \leq_m J$  via  $k$ , rendering  $x \in J$  undecidable.  $\square$



**Worth stating.** Since the  $h$  and  $k$  utilized above are S-m-n functions, they are 1-1. Therefore *all* reducibilities that we have effected above are 1-reducibilities,  $\leq_1$ .



**0.0.8 Theorem.** *None of the sets in 0.0.7 are semi-recursive, except  $C$  and  $D$ .*

*Proof.*  $D$  is semi-recursive by the  $\mathcal{P}$ -graphs theorem and substitution. As for  $C$ , we see that  $y \in \text{ran}(\phi_x) \equiv (\exists z)\phi_x(z) = y$ . Its semi-recursive follows from the same theorem via closure properties of  $\mathcal{P}_*$ .

We now turn to those sets listed in 0.0.7, which we have not already proved to be non c.e. in the proof of said theorem.

(1)  $A = \{x : \phi_x \text{ is a constant function}\}$ .

The obvious approach, that is, badly imitating 0.0.6 and modifying  $(*)$  to read

$$\psi(x, y) = \begin{cases} 0 & \text{if } \phi_x(x) \uparrow \\ \uparrow & \text{if } \phi_x(x) \downarrow \end{cases}$$

will not work. The *intuitive* reason is that if we try to compute this new  $\psi$  in the obvious way, given  $x$  and  $y$  we will ignore  $y$ , and will decode  $x$  to obtain the machine it denotes,  $M$ . We will run  $M$  on input  $x$  and will output and stop everything precisely if  $M$  is in an infinite loop, which is precisely if  $\phi_x(x) \uparrow$ .

Otherwise ( $\phi_x(x) \downarrow$ ) we will ensure that the overall computation never halts by getting into a deliberate-loop.

The catch is that this “obvious” way is doomed, for our program—indeed, no program—can test, or even just verify that  $\phi_x(x) \uparrow$ .

The *definitive* reason that this  $\psi$  is not computable is this: If it were, then so would be  $\lambda x.\psi(x, x)$ . But the domain of the latter is  $\bar{K}$ . Impossible, because this set is not the domain of any partial recursive function.

**Pause.** Why “definitive”? Isn’t the intuitive reason (of the uncomputability of  $\psi$ ) enough? ◀

No. The intuition only *warns* and *guides*; it does not *prove*. After all, the suggested “program that did not work” was just *one* suggested, and “obvious”, program to compute  $\psi$ .

Why can it not be the case that a future programmer might come up with a really clever and non obvious URM that computes  $\psi$ ?

Precisely because we got the definitive answer *mathematically*: There can be *no* such a URM, now or ever; it does not exist.

OK, here is how to do it right: We want to build a partial recursive  $\psi$  such that

$$\psi(x, y) = \begin{cases} 0 & \text{if } \phi_x(x) \uparrow \\ \text{not a constant result} & \text{if } \phi_x(x) \downarrow \end{cases}$$

In view of the above remarks, we cannot use the condition “ $\phi_x(x) \uparrow$ ” outright as the top condition, so we will *approximate* it with “ $\phi_x(x)$  does not *converge*—synonymous for “halt”—in  $\leq y$  steps”.

Note that for a “large” (number of steps)  $y$ , the casual (and impatient) observer will consider a computation for  $\phi_x(x)$ , which is still going, as divergent.

So we finally define

$$\psi(x, y) = \begin{cases} 0 & \text{if } \phi_x(x) \text{ does not converge in } \leq y \text{ steps} \\ \uparrow & \text{if } \phi_x(x) \downarrow \end{cases} \quad (i)$$

This  $\psi$  is computable! Let us see why, intuitively at first. We program as follows: Given inputs  $x$  and  $y$ . We call the URM  $M$ , coded by  $x$ , on input  $x$ . If  $M$  has not stopped after  $y$  steps of its computation, then we print 0 and stop everything. In the contrary case—that is, the call to  $M$  with input  $x$  stopped within  $\leq y$  steps—we deliberately enter an infinite loop.

Mathematically, (i) can be rewritten as

$$\psi(x, y) = \begin{cases} Z(y) & \text{if } \Phi_x(x) > y \\ \emptyset(y) & \text{if } \Phi_x(x) \leq y \end{cases} \quad (ii)$$

where  $Z = \lambda y.0$ . Since the conditions are recursive, we have, by Exercise 27 of Problem Set #1, that  $\psi \in \mathcal{P}$ .\*

By the S-m-n theorem, we have a primitive recursive  $\sigma$  such that, for all  $x$  and  $y$ ,

$$\phi_{\sigma(x)}(y) = \begin{cases} Z(y) & \text{if } \Phi_x(x) > y \\ \emptyset(y) & \text{if } \Phi_x(x) \leq y \end{cases} \quad (iii)$$

Let us now consider the two cases below for any fixed  $x$ :

Case 1:  $\phi_x(x) \uparrow$ . Then  $\Phi_x(x) > y$  is true, for this  $x$  and *all*  $y$ . The top case applies. That is:

$$\phi_{\sigma(x)} = \lambda y.0 \quad (iv)$$

Case 2:  $\phi_x(x) \downarrow$ . Let  $y_0$  be smallest such that  $\Phi_x(x) \leq y_0$ . That is

$$\text{For } y = 0, 1, \dots, y_0 - 1, \text{ we have } \neg \Phi_x(x) \leq y$$

---

\*One is normally less pedantic and rather than explicit function calls  $Z(y)$  and  $\emptyset(y)$  writes 0 and  $\uparrow$  respectively.



In this case

$$\phi_{\sigma(x)} = \overbrace{(0, 0, \dots, 0)}^{y_0 \text{ zeros}} \quad (v)$$

where in (v) we have denoted the finite function

$$f(y) = \text{if } x = 0 \vee x = 1 \vee \dots \vee x = y_0 - 1 \text{ then } 0 \text{ else } \emptyset(y)$$

as the finite sequence of its outputs. Of course,  $f \in \mathcal{P}$ .

We summarize what cases 1 and 2 say in (iv) and (v):

$$\phi_{\sigma(x)} = \begin{cases} \lambda y.0 & \text{if } \phi_x(x) \uparrow \\ \overbrace{(0, 0, \dots, 0)}^{y_0 \text{ zeros}} & \text{if } \phi_x(x) \downarrow \end{cases} \quad (\dagger)$$

Given that the function in the bottom case is *not* a constant function, we immediately have  $\sigma(x) \in A$  iff  $x \in \bar{K}$ , or  $\bar{K} \leq_m A$  as needed.

(2)  $B = \{x : \phi_x \text{ is total}\} = \{x : \phi_x \in \mathcal{R}\}.$

We may reuse (†) immediately above, since the top case is total while the bottom case is nontotal. Thus,  $\sigma(x) \in B$  iff  $x \in \bar{K}$ , or  $\bar{K} \leq_m B$  as needed.

(3)  $G = \{x : \text{dom}(\phi_x) \text{ is infinite}\}.$

We may reuse (†) since the top case has infinite domain while the bottom case has finite domain. Thus,  $\sigma(x) \in G$  iff  $x \in \bar{K}$ , or  $\bar{K} \leq_m G$  as needed.

(4)  $J = \{x : \text{ran}(\phi_x) \text{ is infinite}\}.$

We cannot reuse (†) here as both the top and bottom cases have finite ranges. We work entirely analogously to (ii) above, and define, for all  $x, y$ ,

$$\chi(x, y) = \begin{cases} y & \text{if } \Phi_x(x) > y \\ \emptyset(y) & \text{if } \Phi_x(x) \leq y \end{cases}$$

As  $\chi$  is defined from partial recursive functions by recursive cases, it is in  $\mathcal{P}$ . By S-m-n we have a primitive recursive  $\tau$  such that, for all  $x, y$ ,

$$\phi_{\tau(x)}(y) = \begin{cases} y & \text{if } \Phi_x(x) > y \\ \emptyset(y) & \text{if } \Phi_x(x) \leq y \end{cases}$$

A similar analysis as above shows readily that

If  $\phi_x(x) \uparrow$ , then  $\phi_{\tau(x)} = \lambda y.y$ , while if  $\phi_x(x) \downarrow$ , then

$$\phi_{\tau(x)} = (0, 1, \dots, y_0 - 1)$$

a finite function displayed as a sequence of outputs, where  $y_0$  is smallest  $y$  such that  $\Phi_x(x) \leq y$ .

Thus,

$$\phi_{\tau(x)} = \begin{cases} \lambda y.y & \text{if } \phi_x(x) \uparrow \\ (0, 1, \dots, y_0 - 1) & \text{if } \phi_x(x) \downarrow \end{cases}$$

and therefore  $\tau(x) \in J$  iff  $x \in \bar{K}$ , or  $\bar{K} \leq_m J$  as needed.  $\square$

The techniques used so far are unified in the results that we develop below. First an extension of the technique we used directly above (and in the case  $\{x : \phi_x \text{ is a constant}\}$ .) The following theorem is the contribution of several people (Rice, Myhill, Shapiro, McNaughton). First a definition.

**0.0.9 Definition. (Finite Functions)** A number theoretic function  $f$  is finite iff  $\text{dom}(f)$  is a finite set.  $\square$

**0.0.10 Theorem.** Given  $A = \{x : \phi_x \in \mathcal{C}\}$ , where  $\mathcal{C} \subseteq \mathcal{P}$ . Suppose that some  $f \in \mathcal{C}$  has no finite subfunction  $\xi$  —i.e.,  $\xi \subseteq f$ — that is a member of  $\mathcal{C}$ . Then  $\overline{K} \leq_m A$ .

*Proof.* The idea is to find, using the S-m-n theorem, an  $h \in \mathcal{PR}$  such that

$$\phi_{h(x)} = \begin{cases} f & \text{if } \phi_x(x) \uparrow \\ \xi \text{ (some finite subfunction of } f) & \text{if } \phi_x(x) \downarrow \end{cases} \quad (1)$$

If this succeeds, then

$$h(x) \in A \iff \phi_{h(x)} \in \mathcal{C} \iff \begin{array}{c} \phi_{h(x)} = f \iff x \in \overline{K} \\ \uparrow \\ \xi \notin \mathcal{C} \end{array}$$

Thus,  $\overline{K} \leq_m A$ .

Now to justify (1) we straightforwardly generalize the technique from 0.0.8, case (4). Thus we define, for all  $x, y$ ,

$$\chi(x, y) = \begin{cases} f(y) & \text{if } \Phi_x(x) > y \\ \emptyset(y) & \text{if } \Phi_x(x) \leq y \end{cases}$$

This is a definition by recursive cases, thus  $\chi \in \mathcal{P}$ . By the S-m-n theorem we have an  $h \in \mathcal{PR}$  such that

$$\phi_{h(x)}(y) = \begin{cases} f(y) & \text{if } \Phi_x(x) > y \\ \uparrow & \text{if } \Phi_x(x) \leq y \end{cases} \quad (2)$$

Let us now consider the two cases:

- (a)  $\phi_x(x) \uparrow$ . Then the top condition of (2) is true for all  $y$ , thus  $\phi_{h(x)} = f$  in this case.
- (b)  $\phi_x(x) \downarrow$ . Let  $y_0$  be **the smallest  $y$ -value such that the bottom condition in (2) holds**. Assume first that  $y_0 \geq 1$ . Thus, for  $y = 0, 1, \dots, y_0 - 1$ , we have that  $\Phi_x(x) > y$  holds, and therefore

$$\phi_{h(x)} \text{ is the finite array } \xi \text{ (subfunction of } f): f(0), \dots, f(y_0 - 1)$$

If  $y_0 = 0$ , then the bottom condition holds for all  $y$ , thus  $\phi_{h(x)} = \emptyset$ . We take  $\xi = \emptyset \subseteq f$ .

We have verified (1).  $\square$

**0.0.11 Theorem. (The Rice Lemma)** *Given a complete index set  $A = \{x : \phi_x \in \mathcal{C}\}$ —where  $\mathcal{C} \subseteq \mathcal{P}$ . If some  $f \in \mathcal{C}$  has an extension  $g \in \mathcal{P} - \mathcal{C}$ , then  $\bar{K} \leq_m A$ .*

*Proof.* Let  $\phi_m \in \mathcal{C}$  and  $\phi_n \notin \mathcal{C}$ , where  $\phi_m \subseteq \phi_n$ .<sup>†</sup> The plan is to prove that a primitive recursive  $h$  exists such that

$$\phi_{h(x)} = \begin{cases} \phi_m & \text{if } \phi_x(x) \uparrow \\ \phi_n & \text{if } \phi_x(x) \downarrow \end{cases} \quad (1)$$

As we have already observed, to avail ourselves of the “definition by positive cases” technique, the top case must be the “otherwise”. But if so, we cannot allow, in general, an “output” other than “ $\uparrow$ ”.

Thus, once again, we will approximate the condition  $\phi_x(x) \uparrow$ .

$$\chi(x, y) = \begin{cases} \phi_m(y) & \text{if } \phi_x(x) \text{ does not converge before } \phi_m(y) \text{ does} \\ \phi_n(y) & \text{if } \phi_x(x) \text{ converges before } \phi_m(y) \text{ does} \end{cases}$$

Is  $\chi$  computable? Intuitively, it is. Here is how: Let  $H$  be a URM for verifying  $\phi_x(x) \downarrow$ ,  $M$  a URM that computes  $\phi_m$ , and  $N$  a URM for  $\phi_n$ . Let  $x$  and  $y$  be given.

We run  $H$  on input  $x$ , and  $M$  on input  $y$  in parallel. If  $M$  halts but  $H$  is still running, then we print  $\phi_m(y)$  and stop everything.



**Worth noting.** If each of  $H$  and  $M$  loops for ever, then the top condition is valid; we *correctly* output  $\phi_m(y)$  in this case (we “output” nothing, which is precisely what the call  $\phi_m(y)$  outputs).



If, on the other hand,  $H$  halts before  $M$  does, then we abort  $M$  and call  $N$  on input  $y$  in order to (if convergence is achieved) output  $\phi_n(y)$ .

The mathematical reason for the computability of  $\chi$  is based on the above informal description.

The content of the above  $\diamond$ -comment is the “otherwise”; thus we achieve a definition by positive cases:

$$\chi'(x, y) = \begin{cases} \phi_m(y) & \text{if } (\exists z)(\Phi_m(y) = z \wedge \Phi_x(x) \geq z) \\ \phi_n(y) & \text{if } (\exists z)(\Phi_x(x) = z \wedge \Phi_m(y) > z) \\ \uparrow & \text{otherwise} \end{cases}$$

Since the above is a definition by positive cases—recall that  $\Phi_i(x) = w$ ,  $\Phi_i(x) > w$ , and  $\Phi_i(x) \geq w$  are (primitive recursive)— $\chi' \in \mathcal{P}$ .

**Pause.** But is  $\chi = \chi'$ ? ◀

Yes. The top condition for  $\chi'$  says “ $\phi_x(x)$  does not converge before  $\phi_m(y)$  does”—this is the case of  $\phi_m(y) \downarrow$ , where  $\phi_x(x)$  may or may not converge. The

<sup>†</sup>Cf. p. ??.

case of  $\phi_m(y) \uparrow$  and  $\phi_x(x) \uparrow$  is covered by the “otherwise” as we already have remarked, noticing that both the top and middle cases now fail. The middle condition says “ $\phi_x(x)$  converges *before*  $\phi_m(y)$  does”.

By the S-m-n theorem there is an  $h \in \mathcal{PR}$  such that, for all  $x$  and  $y$ ,

$$\phi_{h(x)}(y) = \begin{cases} \phi_m(y) & \text{if } (\exists z)(\Phi_m(y) = z \wedge \Phi_x(x) \geq z) \\ \phi_n(y) & \text{if } (\exists z)(\Phi_x(x) = z \wedge \Phi_m(y) > z) \\ \uparrow & \text{otherwise} \end{cases}$$

We can now verify that we have (1) above. So, fix an  $x$ :

- Let  $\phi_x(x) \uparrow$ . Then  $\Phi_x(x) \geq z$  is true for all  $z$ , hence **the middle case cannot apply**.
  - (a) If we have  $\phi_m(y) \downarrow$ , then  $(\exists z)\Phi_m(y) = z$ , thus the top condition is true and  $\phi_{h(x)}(y) = \phi_m(y)$ .
  - (b) If we have  $\phi_m(y) \uparrow$ , then  $\Phi_m(y) = z$  is false for all  $z$ , thus only the “otherwise” applies. We have, once more  $\phi_{h(x)}(y) = \phi_m(y)$ .

*For short,  $\phi_{h(x)} = \phi_m$  in this case.*

- Let  $\phi_x(x) \downarrow$ . Let  $z$  be *smallest* such that  $\Phi_x(x) = z$ . Now fix a  $y$ .
  - (i) If  $\Phi_m(y) \leq z$ , then the top case holds, thus  $\phi_{h(x)}(y) = \phi_m(y)$ . But  $\phi_m(y) \downarrow$  (why?), thus, by  $\phi_m \subseteq \phi_n$ , we have  $\phi_m(y) = \phi_n(y)$ . Therefore  $\phi_{h(x)}(y) = \phi_n(y)$ .
  - (ii) If  $\neg\Phi_m(y) \leq z$ , then the middle case holds, and again  $\phi_{h(x)}(y) = \phi_n(y)$ .

*For short,  $\phi_{h(x)} = \phi_n$  in this case.*

This establishes (1), and hence the equivalences  $h(x) \in A$  iff  $\phi_{h(x)} \in \mathcal{C}$  iff (recall:  $\phi_n \notin \mathcal{C}$ )  $\phi_{h(x)} = \phi_m$  iff  $x \in \bar{K}$ . That is,  $\bar{K} \leq_m A$  via  $h$ .  $\square$

**0.0.12 Corollary.** *Given a complete index set  $A = \{x : \phi_x \in \mathcal{C}\}$ —where  $\mathcal{C} \subseteq \mathcal{P}$ . If some  $f \in \mathcal{C}$  has an extension  $g \in \mathcal{P} - \mathcal{C}$ , then  $A$  is not c.e.*

**0.0.13 Corollary. (The Theorem of Rice)** A complete index set  $A = \{x : \phi_x \in \mathcal{C}\}$  is recursive iff it is trivial, **meaning that either**  $A = \emptyset$  **or**  $A = \mathbb{N}$ .

*Proof.* The *if* part is immediate since, in fact,  $\emptyset$  and  $\mathbb{N}$  are primitive recursive. As for the *only if*, say  $A$  is recursive. Then  $A$  and  $\mathbb{N} - A$  (or  $\bar{A}$ ), that is,

$$\{x : \phi_x \in \mathcal{P} - \mathcal{C}\}$$

are both c.e.

We consider two cases. First, let  $\emptyset^\ddagger \in \mathcal{C}$ . Since  $A$  is semi-recursive, 0.0.12 yields that every computable extension of  $\emptyset$  is in  $\mathcal{C}$ . Thus  $\mathcal{C} = \mathcal{P}$  and hence  $A = \mathbb{N}$ .

Second, let  $\emptyset \in \mathcal{P} - \mathcal{C}$ . As above, since  $\mathbb{N} - A$  is c.e., 0.0.12 yields that every computable extension of  $\emptyset$  is in  $\mathcal{P} - \mathcal{C}$ . That is,  $\mathcal{P} - \mathcal{C} = \mathcal{P}$  and hence  $\mathbb{N} - A = \mathbb{N}$ . Therefore,  $A = \emptyset$ .  $\square$

**0.0.14 Corollary.** Let the complete index set  $A = \{x : \phi_x \in \mathcal{C}\}$  be c.e. Then  $f \in \mathcal{C}$  iff some finite subfunction of  $f$  is in  $\mathcal{C}$ .

*Proof.* The *only if* is by 0.0.10, for otherwise  $\bar{K} \leq_m A$ , contradicting the assumption. The *if* is by 0.0.11, for if  $\mathcal{C} \ni \xi \subseteq f$ , then  $f \in \mathcal{C}$ .  $\square$

**0.0.15 Example.** We look back to 0.0.7. We see at once by application of Rice's theorem that each of the sets  $A$ ,  $B$ , and  $E-J$  are not recursive.

Each of them is a nontrivial complete index set. For example, the set of constants  $\mathcal{C}$  is not equal to either  $\emptyset$  or  $\mathcal{P}$ , for, on one hand, computable constant functions exist (!), such as  $\lambda x.380$ , or  $\lambda x.0$ , and, on the other hand, not every computable function is a constant; for example,  $\lambda xy.x + y$ ,  $\lambda x.x$ , etc. Thus,  $\emptyset \neq \mathcal{C} \neq \mathbb{N}$ .

Similarly one shows all of  $E-J$  to be nontrivial.

The sets  $C$  and  $D$  are not complete index sets so the theorem of Rice does not help. One can employ either the technique of 0.0.7 or direct diagonalization.  $\square$

**0.0.16 Example.** We have seen already that every computable function has infinitely many  $\phi$ -indices by arguing the case via URM programs. Here is a "high level" approach: Let  $f \in \mathcal{P}$ . Then  $\emptyset \neq \{x : \phi_x = f\} \neq \mathbb{N}$ . By Rice's theorem,  $\{x : \phi_x = f\}$  is not recursive, hence must be infinite (every finite set is primitive recursive).  $\square$



**0.0.17 Example.** But how about  $K$ ? Is  $K = \{x : \phi_x(x) \downarrow\}$  a complete index set? That is, is there a  $\mathcal{C} \subseteq \mathcal{P}$  such that  $K = \{x : \phi_x \in \mathcal{C}\}$ ? We will answer this negatively later.  $\square$



The Rice Lemma 0.0.11 can help in easily establishing non semi-recursiveness. Let us revisit 0.0.8.

$\ddagger$ The empty function in this context

---

**0.0.18 Example.** Look at  $E = \{x : \text{dom}(\phi_x) = \emptyset\}$ . For convenience let us set

$$\mathcal{C} \stackrel{\text{Def}}{=} \{f \in \mathcal{P} : \text{dom}(f) = \emptyset\}$$

Note that  $\text{dom}(\emptyset) = \emptyset$ . However,  $\lambda x.0$  extends  $\emptyset$  but is not in  $\mathcal{C}$ —its domain is  $\mathbb{N}$ . By 0.0.12,  $E$  is not c.e.  $\square$

**0.0.19 Example.**  $\{x : W_x = \emptyset\}$  is not c.e. Indeed, This is  $\{x : \text{dom}(\phi_x) = \emptyset\}$ . This sends us to the previous example.

$\{x : W_x \text{ is infinite}\}$  is not c.e. by the finite subfunction test. Indeed,  $\{x : W_x \text{ is infinite}\} = \{x : \text{dom}(\phi_x) \text{ is infinite}\}$ . No finite subfunction of an infinite domain function is in the implied  $\mathcal{C}$ .  $\square$





# Bibliography