Recall:

partial computable functions

total computable functions

$\Big\downarrow$ using these

acceptable languages $\equiv$ computably enumerable languages (also called recursively enumerable)

decidable languages $\equiv$ c.e. in increasing order in the standard enumeration [HW1]

Theorem (Kleene)

$\exists\ \mathcal{U}, C$ partial computable st. $\forall$ partiable computable f_e

$$f_e(n) = \mathcal{U}\left[\mu z.\quad C(e,n,z) = 0\right]$$

Theorem (application of diagonalization, a version of halting problem)

There is a partial computable function that is not total computable.

Proof Idea

(Diagonalization)

	0	1	2	3	
f_0	10	15	0	...	
f_1	0	12	0	1	...
f_2	0	0	$\perp$	...	
$\vdots$					

$\boxed{u(\mu z.\ C(n,n,z)=0)+1}$

Consider

$g(n) = f_n(n)+1$ if $f_n(n)$ is defined and $\perp$ otherwise

$\Rightarrow$ "g does not occur in the enumeration"

But g is computable, so g must occur in the enumeration.

In particular, g is undefined

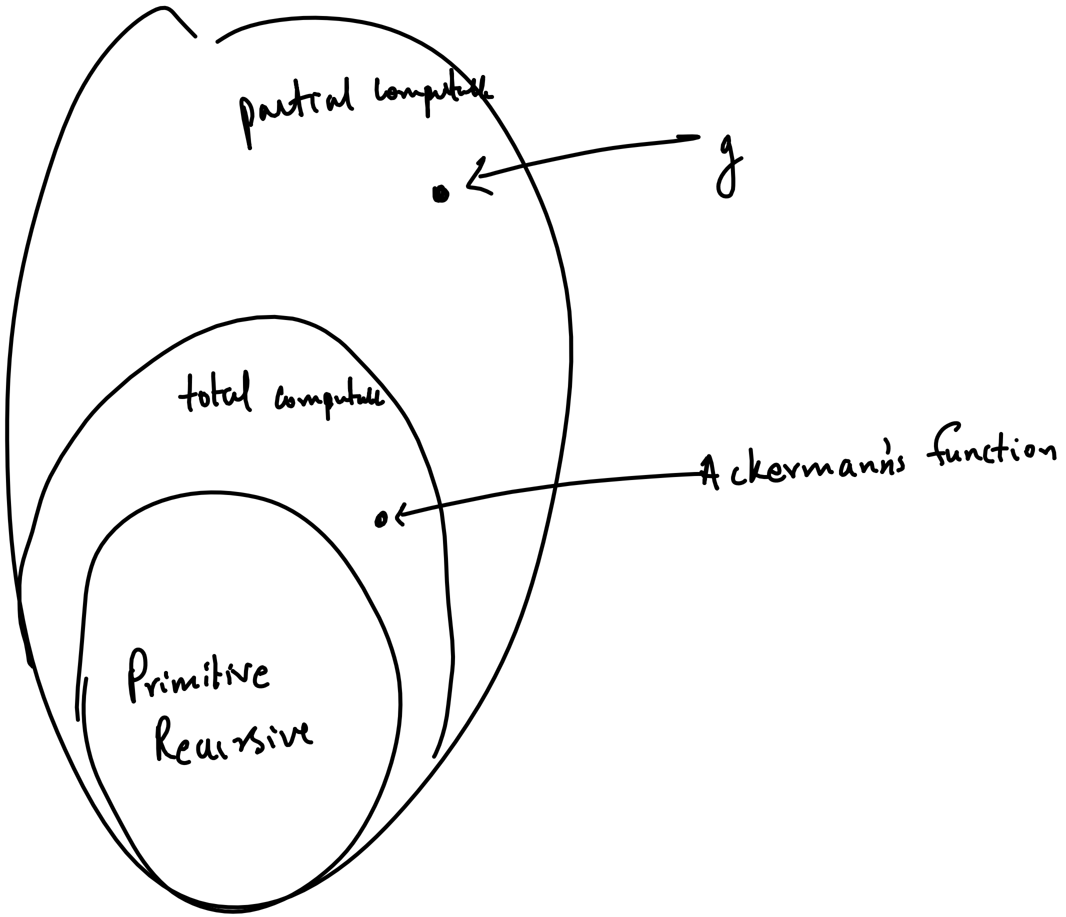
Let $g = f_i$

If g is defined, then

$g(i) = \underline{\underline{f_i}}(i)+1 = g(i)+1$

which is a contradiction.

So $f_i(i) = \perp = g(i)$.


partial computable
g
total computable
Ackermann's function
Primitive
Recursive

Kolmogorov Complexity