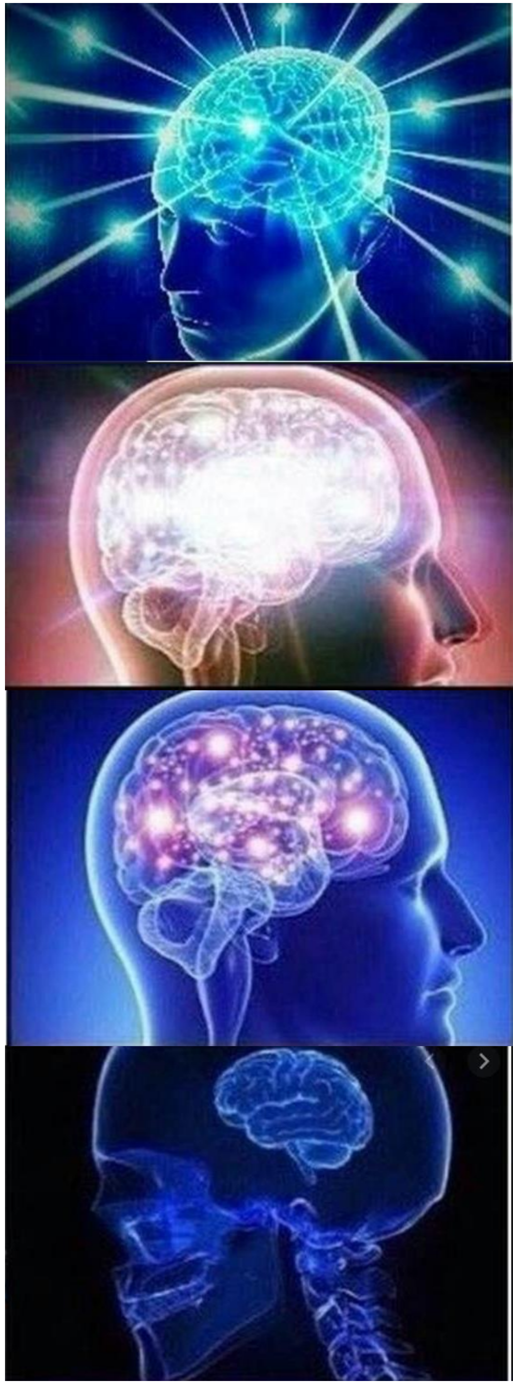
Oracle Machines

Beyond C.E. sets



Co-hosted by Paul

From before

- C.e. sets are those a **computer** can list

- Computable: a computer can list them, and can also list their complements

- The set $K = \{x: \varphi_x(x) \downarrow\}$ is c.e. but not computable

- $\overline{K}$ (the complement of K) is not c.e.

About c.e. sets

- We first defined a set to be c.e. if (means iff) it is empty or the range of a computable function

- We showed that a set is c.e. iff it is the range of a partial computable function

- We also showed that a set is c.e. iff it is the **domain** of a partial computable function

Proof:

Let A be a c.e. set

If A is empty, then A is the domain of the empty function given by the program which doesn't halt on any input

If A is not empty, then it is the range of a computable function, say $A = \{f(0), f(1), f(2), \dots\}$.

Let $\varphi(x) = \mu y[f(y) = x]$. Then $dom(\varphi) = A$

Let’s analyze the last definition of C.E.

- A is c.e. iff it is the domain of a p.c. function f.

- Given any x, if x is in A, then the $f(x)\downarrow$, and if x is not in A, then $f(x)\uparrow$

- So basically, we have a program that will confirm that YES if x is in A, and otherwise the program tells us nothing

Notation

- The domain of φ_e is denoted by W_e

- W_e is the e-th c.e. set

C.E. and ∃

- There is a strong relationship between c.e. and the existential quantifier

- If A is c.e., then for some e, x is in A iff $\exists s\ \varphi_{e,s}(x) \downarrow$

Where, roughly, $\varphi_{e,s}(x) \downarrow$ means that the computation halts within s steps (or stages).

- Note that $\{(e, s, x) : \varphi_{e,s}(x) \downarrow\}$ can be regarded as a relation $R(x_1, x_2, x_3)$

Computable Relations

- Recall, a binary relation over sets *X*, *Y* is a subset of the Cartesian product
$$X \times Y$$

- More generally, an *n*-ary relation over sets $X_1, \dots, X_n$ is a subset of
$$X_1 \times \cdots \times X_n$$

- An *n*-ary relation on $\mathbb{N}$ is one for which $X_1 = \cdots = X_n = \mathbb{N}$

- A relation on $\mathbb{N}$ is computable if it is computable as a set

- We say a relation is c.e. if it is c.e. as a set.

Example

- $R = \{(x, y, z) \in \mathbb{N}^3 : x < y \ and \ z = 2x\}$

We have $R(1,2,2), R(0,3,0), R(10,11,20)$
But $\neg R(0,2,2), \neg R(0,0,0), \neg R(10,11,11)$

Here $\neg$ means negation

- R is clearly computable. There's a program which when given any tuple (a, b, c) it can decide if $R(a, b, c)$ or $\neg R(a, b, c)$

- Note that we can regard relations as Boolean valued functions

- $R_2 = \{(x, e) \in \mathbb{N}^2 \colon \ \varphi_e(x) \downarrow\}$

Not computable (why?)

But it is c.e. because, for any given values a,b, if $R_2(a, b)$ then we can confirm that computably

Special Cases

- Note that a function is a binary relation
- A non-empty subset of X is a unary (1-ary) relation on X.
- There are 0-ary relations (TRUE and FALSE)
- There is the empty relation $\emptyset$ which is the same as FALSE (holds for nothing)

Deeper analysis of $\varphi_e(x)\downarrow$

- We assume s > x and s > e when we write $\varphi_{e,s}(x)\downarrow$
- When we write $\varphi_{e,s}(x)\downarrow = y$, we assume that s is greater than x,e,y
- Recall that the following ternary relation is computable
$$\{(e,s,x) : \varphi_{e,s}(x)\downarrow\}$$

One can prove that:

A relation $R(x, y)$ is c.e. iff there exists a computable relation $C(a, x, y)$

such that for all *x,y*

$$R(x, y) \iff \exists a\, C(a, x, y)$$

The Arithmetical Hierarchy

- We use Σ_1^0 to denote the class of relations (formulas) obtained as $\exists \bar{a}\ C(\bar{a}, \bar{x})$ using some computable relation C

- Π_1^0 denotes the class of relations (formulas) obtained as $\forall \bar{a}\ C(\bar{a}, \bar{x})$ using some computable relation C

- Note that if a set is Σ_1^0 then its complement is Π_1^0 , and vice versa

Going higher

- Π_2^0 denotes the class of relations (formulas) obtained as $\forall\bar{a}\exists\bar{b}\ C\left(\bar{a},\bar{b},\bar{x}\ \right)$ using some computable relation C

Or equivalently $\forall\bar{a}\ D(\bar{a},\bar{x}\ )$ for some Σ_1^0 relation D

- Σ_2^0 denotes the class of relations (formulas) obtained as $\exists\bar{a}\forall\bar{b}\ C\left(\bar{a},\bar{b},\bar{x}\ \right)$ using some computable relation C

In general

- Π^0_{n+1} denotes the class of relations (formulas) obtained as $\forall\bar{a}\ D(\bar{a},\bar{x}\ )$ for some Σ^0_n relation *D*

- Σ^0_{n+1} denotes the class of relations (formulas) obtained as $\exists\bar{a}\ D(\bar{a},\bar{x}\ )$ for some Π^0_n relation *D*

- Note that, for all *n*, $\Sigma^0_n \cup \Pi^0_n \subsetneq \Sigma^0_{n+1} \cap \Pi^0_{n+1}$

- Recall we mentioned that

A relation $R(x, y)$ is c.e. iff there exists a computable relation $C(a, x, y)$ such that for all *x,y*

$$R(x, y) \iff \exists a\, C(a, x, y)$$

- This means that C.E. = Σ_1^0
- BTW, Computable = $\Sigma_0^0 = \Pi_0^0$

The Normal Form Theorem for C.E. Sets

- The following are equivalent:

- A is c.e.
- A is Σ_1^0
- A = W_e for some $e \in \mathbb{N}$

Relative Computability

- We have just seen that C.E. = Σ_1^0
- How about Σ_2^0 ? Or more generally, Σ_{n+1}^0?
- Are they c.e. in some sense w.r.t. some higher level?
- Indeed, it is all about the computable function which enumerates the set

Oracle Machines and Relative Computability

- Imagine a function which is computable but only after giving it certain knowledge
- Imagine its program which allows using the indicator function of some set A (not necessarily computable)
- Such a function is said to be (relatively) computable from A

Turing Reducibility

- A set *S* is said to be Turing reducible to a set *B* ($S \leq_T B$) if the characteristic function of *S* is computable from *B.*
- If $S \leq_T B$ and $S \geq_T B$, then we write $S \equiv_T B$ and say they are Turing equivalent
- $\leq_T$ is a partial order, and $\equiv_T$ is an equivalence relation

Turing Degrees

- The equivalence classes corresponding to $\equiv_T$ are called the Turing degrees (often denoted by bold lowercase **a**, **b**, **c**, ..)
- They are also known as *degrees of unsolvability*
- All computable sets have the same Turing degree (why?)

Structure of the set of Turing Degrees

- Partially ordered but not linearly ordered (there are incomparable degrees)

- There is a smallest Turing degree which is the Turing degree of the empty set (which is also the Turing degree of any computable set)

Notation

- $P_e^A, \boldsymbol{\phi}_e^A, W_e^A$

Program with oracle *A*, p.c. function with oracle *A*, *A*-c.e. set

How to get higher degrees? (the Jump operator)

- Given a set A, consider the halting set with respect to A:

$$A' = K^A = \{x: \boldsymbol{\phi}_x^A(x) \downarrow\} = \{x: x \in W_x^A\}$$

- This set is called the *jump of A* and we have that $A <_T A'$

- $\emptyset' = K$

- $A \equiv_T B$ implies $A' \equiv_T B'$

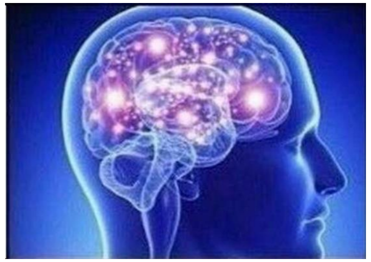
- A' is A-*c.e.* but not A-*computable*

Iterating the jump

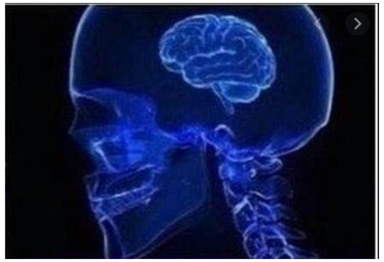
- $\emptyset'', \emptyset''', \ldots$
- $\emptyset^{(2)} = \emptyset''$
- $\emptyset^{(n)}$

- $\deg(\emptyset) = \mathbf{0}$
- $\deg(A)'$ is defined as $\deg(A')$
- $\deg\left(\emptyset^{(n)}\right) = \mathbf{0}^{(n)}$

C.E./ Co-c.e.



Computable level



Other Reducibilities

- Note that Turing reducibility does not distinguish a set from its complement (for any set A, $A \equiv_T \bar{A}$)

- But clearly both sets can be very different in terms of computability properties. Example: K and $\overline{K}$

- Similar properties can be maintained by stronger reducibilities

m-reducibility (many-one reducibility)

- $A \leq_m B$ (A is m-reducible to B) if there exists a computable function f such that: for every $x \in \mathbb{N}$,

$$x \in A \text{ iff } f(x) \in B$$