§1 2020-05-27

§1.1 Universal Functions

Definition 1.1. A binary function $U : \mathbb{N}^2 \to \mathbb{N}$ is said to be universal for the class of computable unary functions if

1. $\forall n,\ U_n : x \mapsto U(n,x)$ is computable. U_n is called a section of U. This is called currying, when you split a function that takes multiple parameters into nested unary functions.
2. $\forall$ unary computable $f : \mathbb{N} \to \mathbb{N}$, $\exists n$ such that $U_n = f$, i.e. $\forall x\ U_n(x) = f(x)$

Note that in this definition U doesn't have to be computable.

Note 1.2. The list of programs is countable.

Theorem 1.3

There is a binary computable function $U : \mathbb{N} \times \mathbb{N} \to N$ such that U is a universal function for all unary computable functions. i.e. one turing machine that can simulate all the others.

Proof. Consider your favorite programming language (YFPL), and enumerate all legal programs, $p_1, p_2, \ldots, p_n, \ldots$

$U(n,x) = p_n(x)$. So U is an "interpreter" while n is the code of the algorithm. □

Note 1.4. "Code" comes from the days of early computability theory because every program could be coded up as a number.

§1.2 Total Computable Universal Function

Does there exist a total computable universal function for the class of total computable unary functions. Total means it will be defined on all inputs, so everything must terminate. No!

Let U be any total computable function of two arguments. Define $d(n) = U(n,n) + 1$. $\forall n,\ d(n) \neq U_n(n)$ so $\forall d \neq U_n$. Can't guarantee all terminating algorithms, or must allow some options to not terminate.

Think about why doesn't this argument work for partial functions? Because $U(n,n)$ might be undefined, in which case $U(n,n) + 1$ is still undefined.

§1.3 Compositional Programming

We want to program Compositionally. If f, g rae computable functions, then $g \circ f$ is also computable. The map that figures out $g \circ f$ should be total computable.

Definition 1.5. Let S be any countable set. A map $\nu : \mathbb{N} \to S$ is called a numbering of S if ν is surjective.

A value of n such that $\nu(n) = s$ is called a code number for s.

Note the indefine article. There can be multiple such n for a given element.

We want to show that for the "right kind" of universal function U, there is a computable function with the following properties.

$$c : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$$
$$\forall p, q, x \in \mathbb{N},\ (U_p \circ U_q)(x) = U(p, U(q, x)) = U(c(p, q), x) = U_{c(p,q)}(x)$$

Note 1.6. A set S is computable if there is a total computable function f, such that $f(n) = 1$ if $n \in S$ and $f(n) = 0$ if $n \notin S$.

Note that computable function doesn't have to be "computable" set. Only total computable function.

Definition 1.7. Let U be a universal computable function. It is called a Godel universal function if $\forall$ binary computable functions V, $\exists$ a total computable unary function $\sigma : \mathbb{N} \to \mathbb{N}$ such that $\forall m, x \in \mathbb{N},\ V(m, x) = U(\sigma(m), x)$ (σ will depend on V).

I stopped taking notes because I realized there was a handout online with detailed notes on today's lecture.

§1.4 Primitive Recursive Functions

Godel. These roughly correspond to a programming language with bounded search. i.e can't use while loops, only loops that run a set number of times.

Fortran for loops for example. Provably terminating.

Ackerman gave an example of a provably terminating function that was not primitive recursive.

This makes sense based on the proof above, whereby it's impossible to produce a universal total computable function for the class of total computable unary functions.

Kleene: PRF + unbounded search gives partial recursive functions. They include the power of while loops. Proved that these are equivalent to turing machines and lambda calculus.

§1.5 Degrees of Unsolvability