ECE664 Lecture 8 Avi Kak

Church's Thesis AND Unsolvability of the Halting Problem

First let's review the notation

$$\psi_{\mathbb{P}}^{(n)}(x_1, \ldots, x_n)$$

the number of input values supplied

value of the output variable y in the terminal snapshot

values for input variables

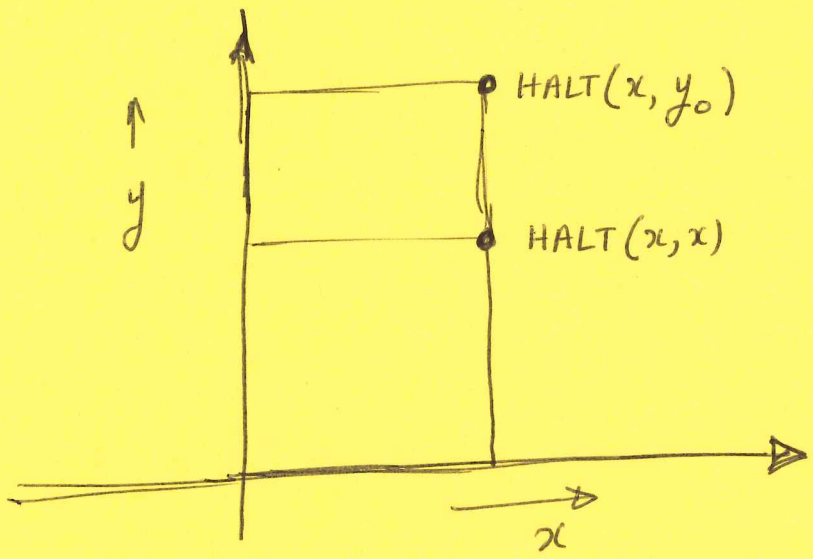
The actual number of input variables used in $\mathbb{P}$ is not limited to n.

It can be $> n$ or $< n$

Let y be the program number for $\mathbb{P}$. That is $y = \#(\mathbb{P})$

Therefore, if we wish, we can execute any $\mathbb{P}$ with the value supplied for just the first input var (and assume all other input vars are set to 0). That is, for any $\mathbb{P}$ we can consider $\psi_{\mathbb{P}}^{(1)}(x)$

The Halting Problem:



We will define a predicate HALT(x, y) in the following manner:

HALT(x, y) is true if $\psi_{\mathbb{P}}^{(1)}(x)$ is defined with $\#(\mathbb{P}) = y$

HALT(x, y) is false if $\psi_{\mathbb{P}}^{(1)}(x)$ is undefined

Think of HALT(x, y) at this time purely in the abstract. Think of it as a possible 2-ary function shown on the left, without at all thinking about how one would go about implementing it.

Theorem: HALT(x, y) is not a computable predicate.

Logically:

HALT(x, y) $\Longleftrightarrow$ program number y eventually halts on input x

Proof: We will prove it by contradiction.

Assume HALT(x,y) is computable.

Then we can write a program

$\mathbb{P}$: [A] IF HALT(X, X) GOTO A

Obviously,

$$\psi_{\mathbb{P}}^{(1)}(x) = \begin{cases} \text{undefined} & \text{if HALT}(x,x) \\ 0 \text{ (defined)} & \text{if } \sim\text{HALT}(x,x) \end{cases}$$

Let's say that y_0 is the program number of $\mathbb{P}$. That is $\boxed{y_0 = \#(\mathbb{P})}$

The above one line program obvious does NOT halt when the value of the input variable X is such that HALT(x,x) is true.

That is HALT(x, y_0) is false when HALT(x, x) is true, and vice versa.

$$\text{HALT}(x, y_0) \Longleftrightarrow \sim\text{HALT}(x,x)$$

This equivalence is obviously true for all x because we could write $\mathbb{P}$ for any arbitrary x for X. So

$$\text{HALT}(y_0, y_0) \Longleftrightarrow \sim\text{HALT}(y_0, y_0)$$

CONTRADICTION

HALT(x, y) is obviously not a computable predicate. That means that there does not exist a program in $\mathcal{S}$ that will compute HALT(x,y) for every x and every y in the "xy-plane".

CHURCH'S THESIS: If a problem involving numbers cannot be solved in $\mathcal{S}$, then it <u>cannot</u> be solved at all.

On the strength of Church's thesis, we conclude that HALT(x, y) is unsolvable.

Representing the State of a Program by a Gödel Number:

In Lecture 3, we talked about the state of a program. A state σ of $\mathcal{P}$ is a list of equations of the form $V = m$ for each variable V of the program.

When we talked about representing a program by a single number, we ordered all the variables of $\mathcal{P}$ in the following manner

$$y \quad X_1 \quad Z_1 \quad X_2 \quad Z_2 \quad X_3 \quad Z_3 \quad \ldots\ldots$$

At the beginning, $y = 0$ and $Z_i = 0$ for $i = 1, 2, \ldots$, and $X_i = x_i$ for $i = 1, 2, \ldots$

We can represent this initial state by the Gödel number:

$$S = [0, x_1, 0, x_2, 0, x_3, \ldots] = \prod_{i=1}^{n} (p_{2i})^{x_i}$$

Let's say that some variable V gets incremented next. Let j be the ordinal index (the position) of this variable in the ordering shown above. The new state of the program will then be

$$S \cdot p_j$$

where p_j is the jth prime. If, on the other hand, some variable is decremented at the very beginning of a program, as to how to change S depends on whether or not that variable was previously non-zero. If that variable was previously non-zero, then S would be divisible by p_j and we go ahead and change S by

$$S / p_j$$

If that variable was previously zero, we leave S unchanged.