

The Parameter Theorem

$$\Phi^{(m+n)}(x_1, \dots, x_m, u_1, \dots, u_n, y) = \Phi^{(m)}(x_1, \dots, x_m, S_m^n(u_1, \dots, u_n, y)).$$

Suppose that the values for u_1, \dots, u_n , and y are fixed.

Then the left side of the equation is a partially computable function of the m arguments x_1, \dots, x_m .

Let the number of the program that computes this function be q . Then we have:

$$\Phi^{(m+n)}(x_1, \dots, x_m, u_1, \dots, u_n, y) = \Phi^{(m)}(x_1, \dots, x_m, q).$$

The parameter theorem tells us that there exists such a q that can be obtained from u_1, \dots, u_n , and y by a primitive recursive function.

November 5, 2009

Theory of Computation
Lecture 15: A Universal Program VII

1

The Parameter Theorem

Let us take a look at the case $n = 1$:

$$\Phi^{(m+1)}(x_1, \dots, x_m, u, y) = \Phi^{(m)}(x_1, \dots, x_m, S_m^1(u, y)).$$

Here, $S_m^1(u, y)$ is the number of a program that receives inputs x_1, \dots, x_m and computes the same value as program number y does on inputs x_1, \dots, x_m , and u .

We can easily obtain $S_m^1(u, y)$ by writing the instruction $X_{m+1} \leftarrow u$ and then appending the program with number y .

This works similarly for any given n , which can be proven by mathematical induction (see page 86 in the textbook).

November 5, 2009

Theory of Computation
Lecture 15: A Universal Program VII

2

Diagonalization and Reducibility

We will now look at two techniques for proving that a set is not recursive or not even r.e.

These techniques are called **diagonalization** and **reducibility**.

The idea behind diagonalization is to conduct a proof by contradiction.

First, we make the assumption that a certain set A can be enumerated in a suitable fashion.

Second, we show that, with the help of the enumeration, we can define an object b with $b \notin A$.

November 5, 2009

Theory of Computation
Lecture 15: A Universal Program VII

3

Diagonalization and Reducibility

For example, we proved Theorem 2.1 using a diagonalization argument to show that $\text{HALT}(x, y)$ or $\{(x, y) \in \mathbb{N}^2 \mid \text{HALT}(x, y)\}$ is not computable.

Here, the set A is the class of unary partially computable functions.

Assertion 1 (A can be enumerated) follows from the fact that \mathcal{L} programs can be enumerated.

For each n , let φ_n be the program with number n .

Then all unary partially computable functions are in the following list: $\Psi_{\varphi_0}^{(1)}, \Psi_{\varphi_1}^{(1)}, \Psi_{\varphi_2}^{(1)}, \dots$

November 5, 2009

Theory of Computation
Lecture 15: A Universal Program VII

4

Diagonalization and Reducibility

We first assumed that $\text{HALT}(x, y)$ were computable and performed a proof by contradiction.

Then we wrote a program \mathcal{P} that computes $\Psi_{\mathcal{P}}^{(1)}$ which involved the (assumedly computable) predicate $\text{HALT}(x, y)$.

Finally, we showed that $\Psi_{\mathcal{P}}^{(1)}$ does not appear in the list $\Psi_{\varphi_0}^{(1)}, \Psi_{\varphi_1}^{(1)}, \Psi_{\varphi_2}^{(1)}, \dots$

From this contradiction we concluded that $\text{HALT}(x, y)$ cannot be computable.

November 5, 2009

Theory of Computation
Lecture 15: A Universal Program VII

5

Diagonalization and Reducibility

The idea was to write \mathcal{P} such that for every x , $\Psi_{\mathcal{P}}^{(1)}(x) \downarrow$ if and only if $\Psi_{\varphi_x}^{(1)}(x) \uparrow$.

In other words,

$$\text{HALT}(x, \#(\mathcal{P})) \Leftrightarrow \sim \text{HALT}(x, x)$$

This means that $\Psi_{\mathcal{P}}^{(1)}(x)$ differs from each function $\Psi_{\varphi_0}^{(1)}, \Psi_{\varphi_1}^{(1)}, \Psi_{\varphi_2}^{(1)}, \dots$ on at least one input value.

$\Psi_{\mathcal{P}}^{(1)}$ cannot be $\Psi_{\varphi_n}^{(1)}$, because

$$\Psi_{\mathcal{P}}^{(1)}(n) \downarrow \text{ if and only if } \Psi_{\varphi_n}^{(1)}(n) \uparrow.$$

November 5, 2009

Theory of Computation
Lecture 15: A Universal Program VII

6

Diagonalization and Reducibility

So $\Psi^{(1)}_e$ does not appear in the list $\Psi^{(1)}_{e_0}, \Psi^{(1)}_{e_1}, \Psi^{(1)}_{e_2}, \dots$, which satisfies Assertion 2 (there is a $b \notin A$).

By this contradiction we can conclude that $\text{HALT}(x, y)$ is not computable.

The term diagonalization is derived from the visualization of this proof technique.

November 5, 2009

Theory of Computation
Lecture 15: A Universal Program VII

7

Diagonalization

$\Psi^{(1)}_{e_0}(0)$	$\Psi^{(1)}_{e_0}(1)$	$\Psi^{(1)}_{e_0}(2)$...
$\Psi^{(1)}_{e_1}(0)$	$\Psi^{(1)}_{e_1}(1)$	$\Psi^{(1)}_{e_1}(2)$...
$\Psi^{(1)}_{e_2}(0)$	$\Psi^{(1)}_{e_2}(1)$	$\Psi^{(1)}_{e_2}(2)$...
...

The elements on the **diagonal** make it impossible for $\Psi^{(1)}_e$ to be any of the $\Psi^{(1)}_{e_n}$.

November 5, 2009

Theory of Computation
Lecture 15: A Universal Program VII

8