

Minimalization

So either $p_n! + 1$ is itself a prime or it is divisible by a prime $> p_n$.

In either case there is a prime q such that $p_n < q \leq p_n! + 1$, which gives us the inequality that we wanted to verify:

$$p_{n+1} \leq p_n! + 1.$$

But now look at the recursion again:

$$p_0 = 0$$

$$p_{n+1} = \min_{t \leq p_n! + 1} [\text{Prime}(t) \ \& \ t > p_n].$$

This is not exactly how we defined recursion.

We should reformulate this definition.

October 6, 2009

Theory of Computation
Lecture 9: A Universal Program I

1

Minimalization

To do so, we define the (obviously) primitive recursive function

$$h(y, z) = \min_{t \leq z} [\text{Prime}(t) \ \& \ t > y]$$

Then we set

$$k(x) = h(x, x! + 1),$$

which is another primitive recursive function.

Then our recursion equations reduce to

$$p_0 = 0$$

$$p_{n+1} = k(p_n),$$

So that we can (finally!) conclude that p_n is a primitive recursive function.

October 6, 2009

Theory of Computation
Lecture 9: A Universal Program I

2

Minimalization

Finally, we want to discuss **minimalization without a bound**.

Let us write

$$\min_y P(x_1, \dots, x_n, y)$$

for the least value of y for which the predicate P is true if there is such a value.

If there is no such value of y , then $\min_y P(x_1, \dots, x_n, y)$ is **undefined**.

(Note the difference with bounded minimalization.)

October 6, 2009

Theory of Computation
Lecture 9: A Universal Program I

3

Minimalization

Obviously, unbounded minimalization of a predicate can produce a function that is not total.

Example:

The function $x - y = \min_z [y + z = x]$ is undefined for $x < y$.

We will see later that there are primitive recursive predicates $P(x, y)$ such that $\min_y P(x, y)$ is a total function which is **not primitive recursive**.

October 6, 2009

Theory of Computation
Lecture 9: A Universal Program I

4

Minimalization

Theorem 7.2: If $P(x_1, \dots, x_n, y)$ is a computable predicate and if $g(x_1, \dots, x_n) = \min_y P(x_1, \dots, x_n, y)$, then g is a partially computable function.

Proof:

The following program computes g :

```
[A] IF P(X1, ..., Xn, Y) GOTO E
    Y ← Y+1
    GOTO A
```

October 6, 2009

Theory of Computation
Lecture 9: A Universal Program I

5

Pairing Functions and Gödel Numbers

How can we code pairs of numbers by single numbers?

Let us define the following primitive recursive function:

$$\langle x, y \rangle = 2^x(2y + 1) - 1.$$

Obviously, $2^x(2y + 1)$ can never be 0, so we have:

$$\langle x, y \rangle + 1 = 2^x(2y + 1).$$

October 6, 2009

Theory of Computation
Lecture 9: A Universal Program I

6

Pairing Functions and Gödel Numbers

$$\langle x, y \rangle + 1 = 2^x(2y + 1).$$

If z is any given number, there is a unique solution x, y to the equation

$$\langle x, y \rangle = z.$$

x is the largest number such that $2^x \mid (z + 1)$, and y is the solution of the equation

$$2y + 1 = (z + 1)/2^x.$$

This equation has a unique solution because $(z + 1)/2^x$ must be odd

(if it were even, we could have chosen a greater x).

October 6, 2009

Theory of Computation
Lecture 9: A Universal Program I

7

Pairing Functions and Gödel Numbers

$$\langle x, y \rangle + 1 = 2^x(2y + 1).$$

Examples:

$$\langle x, y \rangle = 41$$

$$x = 1$$

$$y = 10$$

$$\langle x, y \rangle = 31$$

$$x = 5$$

$$y = 0$$

$$\langle x, y \rangle = 36$$

$$x = 0$$

$$y = 18$$

$$\langle x, y \rangle = 0$$

$$x = 0$$

$$y = 0$$

October 6, 2009

Theory of Computation
Lecture 9: A Universal Program I

8

Pairing Functions and Gödel Numbers

This way the equation $\langle x, y \rangle = z$ defines functions $x = l(z)$ and $y = r(z)$.

$\langle x, y \rangle = z$ also implies that $x, y < z + 1$, and therefore $l(z) \leq z, r(z) \leq z$.

Then we can write:

$$l(z) = \min_{x \leq z} [(\exists y)_{y \leq z} (z = \langle x, y \rangle)],$$

$$r(z) = \min_{y \leq z} [(\exists x)_{x \leq z} (z = \langle x, y \rangle)],$$

showing that $l(z)$ and $r(z)$ are primitive recursive functions.

It is also true that $\langle x, y \rangle = z \Leftrightarrow x = l(z) \ \& \ y = r(z)$.

October 6, 2009

Theory of Computation
Lecture 9: A Universal Program I

9