Minimalization

Example 15: \( R(x, y) \)

\( R(x, y) \) is the remainder when \( x \) is divided by \( y \).
We can also write \( R(x, y) = x \mod y \) ("modulo").
Obviously, it is true that
\[
\frac{x}{y} = \left\lfloor \frac{x}{y} \right\rfloor + \frac{R(x, y)}{y}
\]
Therefore, we can write:
\[
R(x, y) = x - \left( y \cdot \left\lfloor \frac{x}{y} \right\rfloor \right)
\]
This shows that \( R(x, y) \) is primitive recursive.
Note that \( R(x, 0) = x \).

Minimalization

Example 16: \( p_n \)

Here, for \( n > 0 \), \( p_n \) is the \( n \)-th prime number
(in order of size).
In order to make \( p_n \) a total function, we set \( p_0 = 0 \).
Then we have:
\[
\begin{align*}
  p_0 & = 0 \\
p_1 & = 2 \\
p_2 & = 3 \\
p_3 & = 5 \\
p_4 & = 7
\end{align*}
\]

Minimalization

Now consider the following recursion equations:
\[
\begin{align*}
p_0 & = 0 \\
p_{n+1} & = \min_{t \leq p_n! + 1} \{ \text{Prime}(t) \land t > p_n \}.
\end{align*}
\]
To see that these equations are correct, we must verify the following inequality:
\[
p_{n+1} \leq p_n! + 1.
\]
Note that for \( 0 < i \leq n \) we have:
\[
\frac{p_n! + 1}{p_i} = \frac{p_n!}{p_i} + \frac{1}{p_i} = K + \frac{1}{p_i},
\]
where \( K \) is an integer.
Why is \( p_n! \) always divisible by \( p_i \)?

Example: \( n = 4, i = 2 \)
Then \( p_4! = 1 \cdot 2 \cdot 3 \cdot 4 = 24 \), and \( p_3 = 3 \),
so \( \frac{p_4!}{p_3} = \frac{24}{3} = 8 \),
In other words, \( p_i \) is always one of the factors in \( p_n! \).
According to the equation \( (p_n! + 1)/p_i = K + 1/p_i \),
\( p_n! + 1 \) is not divisible by any of the primes \( p_1, \ldots, p_n \).

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:
\[
\begin{align*}
p_0 & = 0 \\
p_{n+1} & = \min_{t \leq p_n! + 1} \{ \text{Prime}(t) \land t > p_n \}.
\end{align*}
\]
This is not exactly how we defined recursion.
We should reformulate this definition.

Minimalization

To do so, we define the (obviously) primitive recursive function
\[
h(y, z) = \min_{t \geq y} \{ \text{Prime}(t) \land t > y \}
\]
Then we set
\[
k(x) = h(x, x! + 1),
\]
which is another primitive recursive function.
Then our recursion equations reduce to
\[
\begin{align*}
p_0 & = 0 \\
p_{n+1} & = k(p_n).
\end{align*}
\]
So that we can (finally!) conclude that \( p_i \) is a primitive recursive function.
Finally, we want to discuss **minimalization without a bound**.

Let us write

\[ \min_y P(x_1, \ldots, x_n, y) \]

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 \( \min_y P(x_1, \ldots, x_n, y) \) is **undefined**.

(Note the difference with bounded 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**.

---

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

**Proof:**

The following program computes \( g \):

[A] IF \( P(X_1, \ldots, X_n, Y) \) GOTO E

\[ Y \leftarrow Y + 1 \]

GOTO A

---

**Pairing Functions and Gödel Numbers**

How can we code pairs of numbers by single numbers?

Let us define the following primitive recursive function:

\[ (x, y) = 2^y(2y + 1) - 1. \]

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

\[ (x, y) + 1 = 2^y(2y + 1). \]