Pairing Functions and Gödel Numbers

How can we code pairs of numbers by single numbers?

Let us define the following primitive recursive function:

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

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

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

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 \(2^x | (z + 1)\), and \(y\) is the solution of the equation

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

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

\[
\langle l(z), r(z) \rangle = z;
\]

Showing that \(l(z)\) and \(r(z)\) are primitive recursive functions.

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

Theorem 8.1 (Pairing Function Theorem):

The functions \(\langle x, y \rangle\), \(l(z)\) and \(r(z)\) have the following properties:

1. they are primitive recursive;
2. \(l(\langle x, y \rangle) = x; r(\langle x, y \rangle) = y;\)
3. \(l(z), r(z)\) are \(\leq z;\)
4. \(l(z), r(z) \leq z.\)
Pairing Functions and Gödel Numbers

For each \( n \), the function \( [a_1, \ldots, a_n] \) is clearly primitive recursive.

Gödel numbering satisfies the following uniqueness property:

**Theorem 8.2:**
If \( [a_1, \ldots, a_n] = [b_1, \ldots, b_n] \) then \( a_i = b_i \) for \( i = 1, \ldots, n \).

This follows immediately from the fundamental theorem of arithmetic, i.e., the uniqueness of the factorization of integers into primes.

However, it is important to note that \( [a_1, \ldots, a_n] = [a_1, \ldots, a_n, 0] \), because for any \( n+1 \), \( p_{n+1} = 1 \).

Actually, we could add any number of 0s to the right end of a sequence without changing its Gödel number.

Since we have \( 1 = 2^0 = 2^03^0 = 2^03^05^0 = \ldots \), it is useful to define 1 as the Gödel number of the empty sequence of length 0.

Pairing Functions and Gödel Numbers

Obviously, adding a zero to the left of the sequence will lead to a Gödel number different from the initial one.

**Examples:**
- \([1, 4] = 2^1 \cdot 3^4 = 162\)
- \([1, 4, 0] = 2^1 \cdot 3^4 \cdot 5^0 = 162\)
- \([0, 1, 4] = 2^0 \cdot 3^1 \cdot 5^4 = 1875\)

We will now define a primitive recursive function \( (x)_i \) so that if \( x = [a_1, \ldots, a_n] \), then \( (x)_i = a_i \).

We set
\[
(x)_i = \min_{t \leq x} (-p_i^{x+t} | x).
\]

Then we define the length \( \text{Lt}(x) \) of the sequence for the Gödel number \( x \):
\[
\text{Lt}(x) = \min_{t \leq x} ((x)_i \neq 0 & (\forall j \leq i \left( (x)_j = 0 \right))).
\]

**Example:** If \( x = 20 = 2^2 \cdot 5^1 = [2, 0, 1] \), then \( (x)_1 = 2, (x)_2 = 0, (x)_3 = 1, (x)_4 = (x)_5 = \ldots 0 \), \( \text{Lt}(x) = 3 \).

Pairing Functions and Gödel Numbers

If \( x > 1 \) and \( \text{Lt}(x) = n \), then \( p_n \) divides \( x \) but no prime greater than \( p_n \) divides \( x \).

Note that \( \text{Lt}([a_1, \ldots, a_n]) = n \) if and only if \( a_n \neq 0 \).

**Theorem 8.3 (Sequence Number Theorem):**

- a. \( ([a_1, \ldots, a_n]) = a_i \) if \( 1 \leq i \leq n \)
- b. \( (\langle x \rangle_1, \ldots, \langle x \rangle_n) = x \) if \( n \geq \text{Lt}(x) \)

After having developed appropriate coding techniques, it will be our goal to enumerate all programs of the language \( \mathcal{L} \).

In other words, each program \( \varphi \) of \( \mathcal{L} \) will receive a number \( \#(\varphi) \) so that the program can be retrieved from its number.

Let us first arrange the variables in the following order:
- \( Y, X_1, Z_1, X_2, Z_2, \ldots \)
- And also the labels:
  - \( A, B, C, D, E, A_1, B_1, C_1, D_1, E_2, A_2, \ldots \)
Coding Programs by Numbers

We write #(V), #(L) for the position of a given variable or label in the appropriate ordering.
For example, #(X2) = 4, #(Z) = 3, #(C2) = 8.

Now let I be an instruction (labeled or unlabeled) of the language $L$.

Then we write $#(I) = \langle a, \langle b, c \rangle \rangle$, where
1. if I is unlabeled, then $a = 0$; if I is labeled $L$, then $a = #(L)$;
2. if the variable V is mentioned in I, then $c = #(V) - 1$;
3. if the statement in I is $V \leftarrow V$, $V \leftarrow V + 1$, or $V \leftarrow V - 1$, then $b = 0, 1, \text{ or } 2$, respectively;
4. if the statement in I is \text{IF } V \neq 0 \text{ GOTO } L' then $b = #(L') + 2$.

Examples:
The number of the unlabeled instruction $X \leftarrow X - 1$ is $\langle 0, \langle 2, 1 \rangle \rangle = \langle 0, 11 \rangle = 22$.
The number of the instruction $[A] X \leftarrow X - 1$ is $\langle 1, \langle 2, 1 \rangle \rangle = \langle 1, 11 \rangle = 45$.

Note that for any given number q there is a unique instruction I with $#(I) = q$. We first calculate $l(q)$.
If $l(q) = 0$, I is unlabeled; otherwise I has the $l(q)$-th label in our list.
To find the variable mentioned in I, we compute $i = r(r(q)) + 1$ and locate the $i$-th variable $V$ in our list.
Then the statement will be $V \leftarrow V$, $V \leftarrow V + 1$, or $V \leftarrow V - 1$, if $l(r(q)) = 0, 1, \text{ or } 2$, respectively; otherwise, it will be the statement \text{IF } V \neq 0 \text{ GOTO } L$, where L is the $j$-th label in our list and $j = l(r(q)) - 2$. 