art.1 Coding Terms

A term is simply a certain kind of sequence of symbols: it is built up inductively from constants and variables according to the formation rules for terms. Since sequences of symbols can be coded as numbers—using a coding scheme for the symbols plus a way to code sequences of numbers—assigning Gödel numbers to terms is not difficult. The challenge is rather to show that the property a number has if it is the Gödel number of a correctly formed term is computable, or in fact primitive recursive.

Variables and constant symbols are the simplest terms, and testing whether \( x \) is the Gödel number of such a term is easy: \( \text{Var}(x) \) holds if \( x \) is \( \#v_i\# \) for some \( i \). In other words, \( x \) is a sequence of length 1 and its single element \( (x)_0 \) is the code of some variable \( v_i \), i.e., \( x = \langle \langle 1, i \rangle \rangle \) for some \( i \). Similarly, \( \text{Const}(x) \) holds if \( x \) is \( \#c_i\# \) for some \( i \). Both of these relations are primitive recursive, since if such an \( i \) exists, it must be \( < x \):

\[
\text{Var}(x) \leftrightarrow (\exists i < x) \ x = \langle \langle 1, i \rangle \rangle \\
\text{Const}(x) \leftrightarrow (\exists i < x) \ x = \langle \langle 2, i \rangle \rangle
\]

Proposition art.1. The relations \( \text{Term}(x) \) and \( \text{ClTerm}(x) \) which hold iff \( x \) is the Gödel number of a term or a closed term, respectively, are primitive recursive.

Proof. A sequence of symbols \( s \) is a term iff there is a sequence \( s_0, \ldots, s_{k-1} = s \) of terms which records how the term \( s \) was formed from constant symbols and variables according to the formation rules for terms. To express that such a putative formation sequence follows the formation rules it has to be the case that, for each \( i < k \), either

1. \( s_i \) is a variable \( v_j \), or
2. \( s_i \) is a constant symbol \( c_j \), or
3. \( s_i \) is built from \( n \) terms \( t_1, \ldots, t_n \) occurring prior to place \( i \) using an \( n \)-place function symbol \( f^k \).

To show that the corresponding relation on Gödel numbers is primitive recursive, we have to express this condition primitive recursively, i.e., using primitive recursive functions, relations, and bounded quantification.

Suppose \( y \) is the number that codes the sequence \( s_0, \ldots, s_{k-1} \), i.e., \( y = \langle \#s_0\#, \ldots, \#s_{k-1}\# \rangle \). It codes a formation sequence for the term with Gödel number \( x \) iff for all \( i < k \):

1. \( \text{Var}(\langle y \rangle_i) \), or
2. \( \text{Const}(\langle y \rangle_i) \), or
3. there is an $n$ and a number $z = \langle z_1, \ldots, z_n \rangle$ such that each $z_l$ is equal to some $(y)_{i'}$ for $i' < i$ and

$$(y)_i = ^{\#} f^n_j (\sim \text{flattened}(z) \sim ^{\#}),$$

and moreover $(y)_{k-1} = x$. (The function flattened$(z)$ turns the sequence $(^\# t_1^#, \ldots, ^\# t_n^#)$ into $^\# t_1, \ldots, t_n^#$ and is primitive recursive.)

The indices $j$, $n$, the Gödel numbers $z_l$ of the terms $t_l$, and the code $z$ of the sequence $\langle z_1, \ldots, z_n \rangle$, in (3) are all less than $y$. We can replace $k$ above with $\text{len}(y)$. Hence we can express “$y$ is the code of a formation sequence of the term with Gödel number $x$” in a way that shows that this relation is primitive recursive.

We now just have to convince ourselves that there is a primitive recursive bound on $y$. But if $x$ is the Gödel number of a term, it must have a formation sequence with at most $\text{len}(x)$ terms (since every term in the formation sequence of $s$ must start at some place in $s$, and no two subterms can start at the same place). The Gödel number of each subterm of $s$ is of course $\leq x$. Hence, there always is a formation sequence with code $\leq x^{\text{len}(x)}$.

For ClTerm, simply leave out the clause for variables.

Problem art.1. Show that the function flattened$(z)$, which turns the sequence $\langle ^\# t_1^#, \ldots, ^\# t_n^# \rangle$ into $^\# t_1, \ldots, t_n^#$, is primitive recursive.

Proposition art.2. The function $\text{num}(n) = ^{\#} n^#$ is primitive recursive.

Proof. We define $\text{num}(n)$ by primitive recursion:

$$\begin{align*}
\text{num}(0) &= ^{\#} 0^#
\text{num}(n + 1) &= ^{\#} t (^\# \sim \text{num}(n) \sim ^{\#})^#.
\end{align*}$$

Photo Credits

Bibliography