Terms.

We fix a first order language and we let
\[ S \]
be its set of symbols. (We may have called this the alphabet previously.)

1. GÖDEL NUMBERING OF SYMBOLS.

\( S \) is the disjoint union of the sets

\[ \text{Punct, Logical, Const, Var, Func, Pred} \]

where Punct is the set of punctuation symbols
\[ (, ) \]
Logical is the set of logical symbols
\[ \sim \ \lor \ \land \ \to \ \leftrightarrow \ \forall \ \exists \]
Const is the set of constant symbols
\[ c_k \]
where the \( k \)s run over a subset of \( \mathbb{N}^+ \) with the property that if \( 1 \leq j < k \) and \( k \) is in this set then so is \( j \);
Var is the set of variable symbols
\[ x_k, \ k \in \mathbb{N}^+ \]
Func is the set of function symbols
\[ f^n_k \]
where the \( (n, k) \)s run over a subset of \( \mathbb{N}^+ \times \mathbb{N}^+ \) with the property that if \( 1 \leq j < k \) and \( (n, k) \) is in this set then so is \( (n, j) \);
Pred is the set of predicate symbols
\[ A^n_k \]
where the \( (n, k) \)s run over a subset of \( \mathbb{N}^+ \times \mathbb{N}^+ \) with the property that if \( 1 \leq j < k \) and \( (n, k) \) is in this set then so is \( (n, j) \).

We let
\[ g : S \rightarrow \mathbb{N} \]
be a function which is univalent and which has the property that the images of each of the six sets of symbols under \( g \) are primitive recursive. It is a simple matter to verify that such functions exist. Such a function is called a GÖDEL NUMBERING OF \( S \).

Let
\[ \text{IsLeft, IsComma, IsRight}; \]
\[ \text{IsNot, IsOr, IsAnd, IsImplies, IsIff, IsForAll, IsExists}; \]
\[ \text{IsConst, IsVar}; \]
\[ \text{IsFunc, IsPred} \]
be the logical functions of one argument defined by requiring that the have value 1 at \( x \in \mathbb{N} \) if and only if \( x = g(s) \) for some \( s \) in the set of symbols corresponding the the name of the function.
We define the function
\[ \text{GetNargs} \]
of one argument with values in \( \mathbb{N} \) at \( x \in \mathbb{N} \) by requiring that \( \text{GetNargs}(x) = n \in \mathbb{N}^+ \) if and only if \( x = g(f^n_k) \) for some \( f^n_k \in \text{Func} \) or \( x = g(A^n_k) \) for some \( A^n_k \in \text{Pred} \) and that \( \text{GetNargs}(x) = 0 \) otherwise.

We define the function
\[ \text{GetIndex} \]
of one argument with values in \( \mathbb{N} \) at \( x \in \mathbb{N} \) by requiring that \( \text{GetIndex}(x) = k \in \mathbb{N}^+ \) if and only if \( x = g(f^n_k) \) for some \( f^n_k \in \text{Func} \) or \( x = g(A^n_k) \) for some \( A^n_k \in \text{Pred} \) and that \( \text{GetNargs}(x) = 0 \) otherwise.

It follows easily that all these functions are primitive recursive.

2. Codes again.

Recall the function
\[ \Gamma : \mathbb{N}^* \to \mathbb{N} \]
defined by setting \( \Gamma(\emptyset) = 0 \) and setting
\[ \Gamma(x) = 2^n \Pi_{i=1}^n \text{Pth}(i)x_i \]
if \( n \in \mathbb{N} \) and \( x = (x_1, \ldots, x_n) \in \mathbb{N}^n \).

We called \( \Gamma(x) \) the code of \( x \). Recall that \( \Gamma \) is univalent with range equal \( \mathbb{N} \).

**Proposition 2.1.** Suppose \( n \in \mathbb{N} \) and \( x \in \mathbb{N}^n \). Then \( \Gamma(x) \) does not exceed \( 2^n \text{Pth}(n)^n \). Moreover, if \( y \) is a subtuple of \( x \) then \( \Gamma(y) \) does not exceed \( \Gamma(x) \) with equality only if \( y = x \).

**Exercise 2.1.** Prove this.

**Definition 2.1.** Recall that \( S^* \) is the set of tuples of symbols. We define
\[ c : S^* \to \mathbb{N} \]
as follows. Suppose \( s \in S^* \). If \( s \) is the empty tuple then \( c(s) = 0 \). If \( n \in \mathbb{N}^+ \) and \( s = (s_1, \ldots, s_n) \) then
\[ c(s) = \Gamma(g(s_1), \ldots, g(s_n)) \]
here \( g(s_i) \) is the Gödel number of \( s_i, i = 1, \ldots, n \). We say, somewhat ambiguously, that \( c(s) \) is the code of \( s \).

We define
\[ C : (S^*)^* \to \mathbb{N} \]
as follows. Suppose \( U \in (S^*)^* \). If \( U \) is the empty tuple then \( C(U) = 0 \). If \( N \in \mathbb{N}^+ \) and \( U = (U_1, \ldots, U_N) \) then
\[ C(U) = \Gamma(c(U_1), \ldots, c(U_N)) \]
We say, somewhat ambiguously, that \( C(U) \) is the code of \( U \).

3. Terms.

**Definition 3.1.** Let
\[ \text{Term} \]
be the logical function of one argument whose value at \( y \in \mathbb{N} \) is 1 if and only if \( y = c(u) \) for some term \( u \).

**Theorem 3.1.** Term is primitive recursive.
3.1. The proof. The remainder of this section is devoted to the proof of this Theorem.

3.1.1. Simple terms. For \( y \in \mathbb{N} \) let

\[
\text{SimpTerm}(y) = (\text{Len}(y) = 1) \land (\text{IsConst}(\text{Cmp}(y, 1)) \lor \text{IsVar}(\text{Cmp}(y, 1))).
\]

Evidently, SimpTerm is primitive recursive and

\[
\text{SimpTerm}(y) = 1 \text{ if and only if for some } u \in S^* \text{ with } y = c(u) \text{ there is } k \in \mathbb{N}^+ \text{ such that } u = (a_k) \text{ or } u = (x_k).
\]

3.1.2. Functional terms. We define the logical functions \( P_1, P_2, P_3, P_4, P_5 \) as well as the functions \( F_1, F_2, F_3, F_4, F_5 \) with arguments as indicated below as follows:

\[
\begin{align*}
P_1(y) &= \text{IsFunc}(\text{Cmp}(y, 1)), \\
P_2(y) &= \text{IsLeft}(\text{Cmp}(y, 2)), \\
P_3(y) &= \text{IsRight}(\text{Cmp}(y, \text{Len}(y))), \\
F_1(y) &= \text{GetNargs}(\text{Cmp}(y, 1)), \\
F_2(y) &= 2^{F_1(y)} P_{\text{th}}(F_1(y)) \text{Len}(y), \\
F_3(z, i) &= \text{Cmp}(z, i), \\
F_4(z, i) &= 1 + i + \sum_{1 < j < i} F_3(z, j), \\
P_5(y, z, i) &= \text{GetSubStr}(y, F_4(z, i) + 1, F_3(z, i)), \\
P_5(y, z, i) &= \text{GetSubStr}(y, F_4(z, i) + 1, F_3(z, i)), \\
P_4(y, z, i) &= \text{IsComma}(\text{Cmp}(y, F_4(z, i))), \\
P_5(y, z, i) &= \text{IsSubStr}(y, z, F_4(z, i) + 1, F_3(z, i))
\end{align*}
\]

for \( y, z, i \in \mathbb{N} \). Note that all these functions are primitive recursive.

Suppose \( n, k \in \mathbb{N}^+, t_i, i = 1, \ldots, n, \) are terms,

\[
(1) \quad u = f_k^n(t_1, \ldots, t_n) \quad \text{and} \quad y = c(u).
\]

We have

\[
|u| = \text{Len}(y) \geq 4
\]

as well as

\[
P_1(y) \land P_2(y) \land P_3(y) = 1
\]

and

\[
n = F_1(y).
\]

Let

\[
z = c(|t_1|, \ldots, |t_n|).
\]

Evidently

\[
0 < z = 2^n \Pi_{i=1}^n P_{\text{th}}(|t_i|) < 2^n p^{\text{Len}(y)} = F_2(y)
\]

where we have set

\[
p = P_{\text{th}}(\text{Len}(y)).
\]

We have

\[
|t_i| = F_3(z, i) \quad \text{for } 1 \leq i \leq n.
\]
Suppose $1 \leq i \leq n$. Let
\[ I_i = 1 + i + \sum_{1 \leq j < i} |t_j| \]
and note that $I_i$ is the index in $u$ of the symbol immediately preceding $t_i$. It follows that
\[ I_i = F_4(z, i), \]
\[ P_4(y, z, i) = 1 \quad \text{if } 1 < i < n, \]
and
\[ t_i = (u_{I_i+1}, \ldots, u_{I_i+|t_i|}). \]
This last equation implies
\[ c(t_i) = P_5(y, z, i). \]
It follows that
\[ (P_1(y) \land P_2(y) \land P_3(y)) \]
\[ (\exists z_{0\leq z<F_5(y)}) \left( \right) \]
\[ (\text{Len}(z) = F_1(y)) \]
\[ \land \]
\[ (\forall i_{1<i<F_5(y)}) P_4(z, i) \]
\[ \land \]
\[ (\forall i_{1\leq i<F_5(y)} (\text{Term}(F_5(y, z, i)) \land P_5(y, z, i)) \]
\[ ) \]
equals 1. Conversely, if this logical function has value 1 then $y$ is the code of a term as in (1).
Finally, if $1 \leq i \leq n$,
\[ \text{Term}(F_5(y, z, i)) = \alpha(\text{Term})(y, F_5(y, z, i)) \]
since $F_5(y, z, i) < y$; this is the case since $t_i$ is a substring of $u$ which is not equal $u$. That Term is primitive recursive follows from the Theorem on “course of values” recursion.