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

$$\text{Punct}, \quad \text{Logical}, \quad \text{Const}, \quad \text{Var}, \quad \text{Func}, \quad \text{Pred}$$

where Punct is the set of punctuation symbols

$$(\qquad , \qquad)$$

Logical is the set of logical symbols

$$\sim \qquad \vee \qquad \wedge \qquad \to \qquad \leftrightarrow \qquad \forall \qquad \exists$$

Const is the set of constant symbols

$$c_k$$

where the ks run over a subset of $\mathbb{N}^+$ with the property that if $1 \le j < k$ and k is in this set then so is j;

Var is the set of variable symbols

$$x_k, \quad k \in \mathbb{N}^+$$

Func is the set of function symbols

$$f_k^n$$

where the (n,k)s run over a subset of $\mathbb{N}^+ \times \mathbb{N}^+$ with the property that if $1 \le j < k$ and (n,k) is in this set then so is (n,j);

Pred is the set of predicate symbols

$$A_k^n$$

where the (n,k)s run over a subset of $\mathbb{N}^+ \times \mathbb{N}^+$ with the property that if $1 \le j < k$ and (n,k) is in this set then so is (n,j).

We let

$$\mathbf{g} : S \to \mathbb{N}$$

be a function which is univalent and which has the property that the images of each of the six sets of symbols under $\mathbf{g}$ 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

$$\text{IsLeft}, \quad \text{IsComma}, \quad \text{IsRight};$$
$$\text{IsNot}, \quad \text{IsOr}, \quad \text{IsAnd}, \quad \text{IsImplies}, \quad \text{IsIff}, \quad \text{IsForAll}, \quad \text{IsExists};$$
$$\text{IsConst}, \quad \text{IsVar};$$
$$\text{IsFunc}, \quad \text{IsPred}$$

be the logical functions of one argument defined by requiring that the have value 1 at $x \in \mathbb{N}$ if and only if $x = \mathbf{g}(s)$ 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 $x = \mathbf{g}(f_k^n)$ for some $f_k^n \in \text{Func}$ or $x = \mathbf{g}(A_k^n)$ for some $A_k^n \in \text{Pred}$ 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 $x = \mathbf{g}(f_k^n)$ for some $f_k^n \in \text{Func}$ or $x = \mathbf{g}(A_k^n)$ for some $A_k^n \in \text{Pred}$ 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

$$\Gamma(x) = 2^n \Pi_{i=1}^n \text{Pth}(i)^{x_i} \quad \text{if } n \in \mathbb{N} \text{ and } x = (x_1, \ldots, x_n) \in \mathbb{N}^n.$$

We called $\Gamma(x)$ **the code of** x. *Recall that* Γ *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

$$\mathbf{c} : S^* \to \mathbb{N}$$

as follows. Suppose $s \in S^*$. If s is the empty tuple then $\mathbf{c}(s) = 0$. If $n \in \mathbb{N}^+$ and $s = (s_1, \ldots, s_n)$ then

$$\mathbf{c}(s) = \Gamma(\mathbf{g}(s_1), \ldots, \mathbf{g}(s_n));$$

here $\mathbf{g}(s_i)$ is the Gödel number of s_i, $i = 1, \ldots, n$. We say, somewhat ambiguously, that $\mathbf{c}(s)$ **is the code of** s.

We define

$$\mathbf{C} : (S^*)^* \to \mathbb{N}$$

as follows. Suppose $U \in (S^*)^*$. If U is the empty tuple then $\mathbf{C}(U) = 0$. If $N \in \mathbb{N}^+$ and $U = (U_1, \ldots, U_N)$ then

$$\mathbf{C}(U) = \Gamma(\mathbf{c}(U_1), \ldots, \mathbf{c}(U_N)).$$

We say, somewhat ambiguously, that $\mathbf{C}(U)$ **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 $y = \mathbf{c}(u)$ 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

$$\mathrm{SimpTerm}(y) = (\mathrm{Len}(y) = 1) \wedge (\mathrm{IsConst}(\mathrm{Cmp}(y,1)) \vee \mathrm{IsVar}(\mathrm{Cmp}(y,1))).$$

Evidently, SimpTerm is primitive recursive and

$$\mathrm{SimpTerm}(y) = 1$$

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

3.1.2. *Functional terms.* We define the logical functions $\mathrm{P}_1, \mathrm{P}_2, \mathrm{P}_3, \mathrm{P}_4, \mathrm{P}_5$ as well as the functions $\mathrm{F}_1, \mathrm{F}_2, \mathrm{F}_3, \mathrm{F}_4, \mathrm{F}_5$ with arguments as indicated below as follows:

$$\begin{aligned}
\mathrm{P}_1(y) &= \mathrm{IsFunc}(\mathrm{Cmp}(y,1)),\\
\mathrm{P}_2(y) &= \mathrm{IsLeft}(\mathrm{Cmp}(y,2)),\\
\mathrm{P}_3(y) &= \mathrm{IsRight}(\mathrm{Cmp}(y,\mathrm{Len}(y))),\\
\mathrm{F}_1(y) &= \mathrm{GetNargs}(\mathrm{Cmp}(y,1)),\\
\mathrm{F}_2(y) &= 2^{\mathrm{F}_1(y)}\mathrm{Pth}(\mathrm{F}_1(y))^{\mathrm{Len}(y)},\\
\mathrm{F}_3(z,i) &= \mathrm{Cmp}(z,i),\\
\mathrm{F}_4(z,i) &= 1 + i + \sum_{1<j<i} \mathrm{F}_3(z,j),\\
\mathrm{P}_5(y,z,i) &= \mathrm{GetSubStr}(y, \mathrm{F}_4(z,i)+1, \mathrm{F}_3(z,i))\\
\mathrm{P}_5(y,z,i) &= \mathrm{GetSubStr}(y, \mathrm{F}_4(z,i)+1, \mathrm{F}_3(z,i)),\\
\mathrm{P}_4(y,z,i) &= \mathrm{IsComma}(\mathrm{Cmp}(y,\mathrm{F}_4(z,i))),\\
\mathrm{P}_5(y,z,i) &= \mathrm{IsSubStr}(y, z, \mathrm{F}_4(z,i)+1, \mathrm{F}_3(z,i))
\end{aligned}$$

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,

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

We have

$$|u| = \mathrm{Len}(y) \geq 4$$

as well as

$$\mathrm{P}_1(y) \wedge \mathrm{P}_2(y) \wedge \mathrm{P}_3(y) = 1$$

and

$$n = \mathrm{F}_1(y).$$

Let

$$z = \mathbf{c}(|t_1|, \ldots, |t_n|).$$

Evidently

$$0 < z = 2^n \Pi_{i=1}^n \mathrm{Pth}(i)^{|t_i|} < 2^n p^{\mathrm{Len}(y)} = \mathrm{F}_2(y)$$

where we have set

$$p = \mathrm{Pth}(\mathrm{Len}(y)).$$

We have

$$|t_i| = \mathrm{F}_3(z,i) \quad \text{for } 1 \leq i \leq n.$$

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

$$I_i = \mathrm{F}_4(z, i),$$
$$\mathrm{P}_4(y, z, i) = 1 \quad \text{if } 1 < i < n,$$

and

$$t_i = (u_{I_i+1}, \ldots, u_{I_i+|t_i|}).$$

This last equation implies

$$\mathbf{c}(t_i) = \mathrm{P}_5(y, z, i).$$

It follows that

$$\begin{aligned}
&(\mathrm{P}_1(y) \wedge \mathrm{P}_2(y) \wedge \mathrm{P}_3(y))\\
&\quad (\exists\, z)_{0<z<\mathrm{F}_2(y)}\Big(\\
&\qquad (\mathrm{Len}(z) = \mathrm{F}_1(y))\\
&\qquad \wedge\\
&\qquad (\forall\, i)_{1<i<\mathrm{F}_1(y)}\mathrm{P}_4(z, i)\\
&\qquad \wedge\\
&\qquad (\forall\, i)_{1\leq i\leq \mathrm{F}_1(y)}(\mathrm{Term}(\mathrm{F}_5(y, z, i)) \wedge \mathrm{P}_5(y, z, i))\\
&\quad \Big)
\end{aligned}$$

equals 1. Conversely, if this logical function has value 1 then y is the code of a term as in (1).

Finally, if $1 \leq i \leq n$,

$$\mathrm{Term}(\mathrm{F}_5(y, z, i)) = \alpha(\Lambda(\mathrm{Term})(y), \mathrm{F}_5(y, z, i))$$

since $\mathrm{F}_5(y, z, i) < y$; 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.