Amin Zolfagharian - 610300118

---

**Problem 1.**

Let $f(n, m, x) = (x+m)^n$. This function is partial recursive since addition and power are both partial recursive functions. By *s-m-n* there exists a computable total function $s$ such that $f(n, m, x) \cong \phi_{s(n,m)}(x)$. This proves the problem:

$$(x+m)^n = f(n, m, x) \cong \phi_{s(n,m)}(x)$$

**Problem 2.**

Since $f$ is a total computable function there exists a standard $URM$-program $F$ that computes $f$. Now we will propose an algorithm for a $URM$ program that computes $h(x)$:
Let $r = \rho(F)$.

Start: T(1, r + 1)
Z(1)
Program: F
J(1, r + 1, End)
S(1)
J(1, 1, Program)
End: Z(1)
S(1)

This program runs over all of numbers, starting from 0. Since $f$ is total then for any $x$, $f(x)$ halts. This program runs $F$ over all numbers and if for some $y$, $f(y) = x$ then gives 1 as output and halts. If there is no such $y$ such that $f(y) = x$ then the program never halts. Now if $h$ is the function for this program we would have:

$$h(x) = \begin{cases} 1 & \textit{if there exists } y \textit{ such that } f(y) = x \iff x \in Ran(x) \\ \uparrow & \textit{if there is no } y \textit{ such that } f(y) = x \iff x \notin Ran(x) \end{cases}$$

This proves the problem.

**Problem 3.**

Let $f(u, v, x) = \phi_u(\phi_v(x))$. Since this function is computable (both $\phi_u$ and $\phi_v$ are computable), then by *s-m-n* theorem, there exists a total and computable $g : \mathbb{N}^2 \to \mathbb{N}$ such that:

$$\phi_u(\phi_v(x)) = f(u, v, x) \cong \phi_{g(u,v)}(x)$$

**Problem 4.**

Let $g(x, y) = \phi_x(f(y))$. Since both $f$ and $\phi_x$ are computable then $g$ is also computable. By *s-m-n* theorem there exists a total computable $k$ such that:

$$\begin{aligned} f(x, y) &\cong \phi_{k(x)}(y) \\ \implies \phi_{k(x)}(y) &= \phi_x(f(y)) \end{aligned}$$

Thus we have:

$$\begin{aligned} y \in W_{k(x)} \iff f(y) \in W_x \iff y \in f^{-1}(W_x) \\ \implies W_{k(x)} = f^{-1}(W_x) \end{aligned}$$

**Problem 5.**

***(i)*** $\beta(J(m, n, q)) = 4(\pi(\pi(m-1, n-1), q-1)) + 3$:

$$\begin{aligned} \pi(3-1, 4-1) = 2^2(2 \times 3 + 1) - 1 = 27 \\ \pi(27, 2-1) = 2^{27}(2 \times 1 + 1) - 1 = 3 \cdot 2^{27} - 1 \\ \implies \beta(J(3, 4, 2)) = 4(3 \cdot 2^{27} - 1) + 3 = 3 \cdot 2^{29} - 1 \end{aligned}$$

***(ii)*** $503 = 4(125) + 3$. Therefore this code belongs to a jump instruction.

$$\begin{aligned} 125 = 2^1(2 \times 31 + 1) - 1 \implies 125 = \pi(1, 31) \\ 1 = 2^1(2 \times 0 + 1) - 1 \implies 1 = \pi(1, 0) \\ \implies 125 = \zeta(2, 1, 32) \\ \implies \beta^{-1}(503) = J(2, 1, 32) \end{aligned}$$

***(iii)*** First we compute the code for each instruction, then we use them to compute the program code:

$$\begin{aligned} \beta(T(3, 4)) = 4(\pi(3-1, 4-1)) + 2 = 4(\pi(2, 3)) + 2 = 4 \cdot 27 + 2 = 110 \\ \beta(S(3)) = 4(3-1) + 1 = 9 \\ \beta(Z(1)) = 4(1-1) = 0 \\ \tau(111, 9, 0) = 2^{110} + 2^{120} + 2^{121} - 1 \end{aligned}$$

***(iv)*** $100 = 2^0 + 2^2 + 2^5 + 2^6 - 1$. Thus $\tau^{-1}(100) = (0, 1, 2, 0)$

$$\begin{aligned} \beta^{-1}(0) = Z(1) \\ \beta^{-1}(1) = S(1) \\ \beta^{-1}(2) = T(1, 1) \\ \beta^{-1}(0) = Z(1) \end{aligned}$$

Thus the program is: $Z(1), S(1), T(1, 1), Z(1)$.

**Problem 6.**

**Kleene's Normal Form**: There exists a total computable function $U : \mathbb{N} \to \mathbb{N}$ such that for any $n \in \mathbb{N}$ There exists a decidable predicate like $T_n(e, \overrightarrow{x}, z)$ such that for any computable function $f : \mathbb{N}^n \to \mathbb{N}$ we have:

$$f(\overrightarrow{x}) = U(\mu_z(T_n(e, \overrightarrow{x}, z)))$$

**Proof**: Let $T_n(e, \overrightarrow{x}, z) = S_n(e, \overrightarrow{x}, l(z), r(z))$ where $e$ is the code of $f$ and $U(x) = l(x)$, where $l$ and $r$ are inverse of $\pi$ such that $x = \pi(l(x), r(x))$. Now if $f(\overrightarrow{x}) \downarrow$, then after $t$ steps the program for $f$ halts for some $t$ and has an output $k$. then for $z = \pi(k, t)$ we have $T_n(e, \overrightarrow{x}, z) = 1$. Also for any $z$ that $T_n(e, \overrightarrow{x}, z) = 1$, value of $l(z)$ is the same. Since it is the output of $f$ over $\overrightarrow{x}$ after $t$ steps such that the computation is over. Thus in this case $f(\overrightarrow{x}) = U(\mu_z(T_n(e, \overrightarrow{x}, z)))$. Now if $f(\overrightarrow{x}) \uparrow$ then $T_n(e, \overrightarrow{x}, z) = 0$ for any $z$. Thus $U(\mu_z(T_n(e, \overrightarrow{x}, z))) \uparrow$. This proves the problem.
The importance of this theorem is that since the characteristic function for $S_n$ is partial recursive then the characteristic function for $T_n$ is also partial recursive, and $U$ is also partial recursive, Therefore $f(\overrightarrow{x})$ can be written with only one usage of $\mu$.