Assignment 5: Recursion Theorem, Strings, Turing Machines

1. (a) Use the Recusion theorem to show that there is a partially computable function f that satisfies the equations

$$f(x, 0) = x + 2$$

$$f(x, 1) = 2f(x, 2x)$$

$$f(x, 2t + 2) = 3f(x, 2t)$$

$$f(x, 2t + 3) = 4f(x, 2t + 1)$$

   (b) Show that f is total.

   (a) We begin by defining a function to which we will apply the recursion theorem. Define

$$F(z, x, y) = \begin{cases} x + 2 & \text{if } y = 0; \\ 2\Phi_z(x, 2x) & \text{if } y = 1; \\ 3\Phi_z(x, y - 2) & \text{if } (\exists t \leq y)(y = 2t + 2); \\ 4\Phi_z(x, y - 2) & \text{if } (\exists t \leq y)(y = 2t + 3) \end{cases} .$$

   Since F is defined using cases and the universal functions, it is partially computable. Thus, by the recursion theorem, there exists a number e so that $\Phi_e(x, y) = F(e, x, y)$. Then by definition of F,

$$\Phi_e(x, 0) = x + 2;$$

$$\Phi_e(x, 1) = \Phi_e(x, 2x)$$

$$\Phi_e(x, 2t + 2) = 3\Phi_e(x, 2t)$$

$$\Phi_e(x, 2t + 3) = 4\Phi_e(x, 2t + 1)$$

   Thus, the partially computable function Φ_e satsifies the required conditions.

   (b) We will show that $f(x, t)$ is total by first showing that it is defined for all even t, then for all odd t. Each of these will be proven by induction. Thus, we begin by showing that for any t, $f(x, 2t+2)$ is defined. The base case is $f(x, 0)$, which is $x + 2$, so it is defined. Assuming we know that $f(x, 2t + 2)$ is defined, then $f(x, 2t + 4)$ is also defined since $f(x, 2t+4) = 3f(x, 2t+2)$. Thus, $f(x, t)$ is defined for all even t.

   We know prove that $f(x, t)$ is defined for all odd t. The base case is $f(x, 1)$. This equals $2f(x, 2x)$, which have proven is defined, since $2x$ is always even. Assuming we know that $f(x, 2t + 1)$ is defined, then $f(x, 2t + 3)$ is also defined since it equals $4f(x, 2t + 1)$. Thus, $f(x, t)$ is defined for all odd t. Thus $f(x, t)$ is defined for all values.

2. Determine the following:

   (a) for the alphabet $\{s_1, s_2\}$, the number of the string: $s_1s_2s_1s_2$;
   (b) for the alphabet $\{s_1, s_2, s_3\}$, the number of the string: $s_3s_2s_3s_1$;
   (c) for the alphabet $\{s_1, s_2\}$, the string with number 100;
   (d) for the alphabet $\{s_1, s_2, s_3, s_4\}$, the string with number 100.

   (a) For the alphabet $\{s_1, s_2\}$, the number of $s_1s_2s_1s_2$ is

   $$(2)^3(1) + (2)^2(2) + (2)(1) + 2 = 20$$

   (b) For the alphabet $\{s_1, s_2, s_3\}$, the number of $s_3s_2s_3s_1$ is

   $$(3)^4(3) + (3)^3(2) + (3)(3) + 1 = 271$$

   (c) To calculate the string in $\{s_1, s_2\}$ with number 100, we must first find the u_i values of 100 relative to 2:

   $$u_0 = 100, u_1 = Q^+(100, 2) = 49, u_2 = Q^+(49, 2) = 24,$$

   $$u_3 = Q^+(24, 2) = 11, u_4 = Q^+(11, 2) = 5, u_5 = Q^+(5, 2) = 2, u_6 = Q^+(2, 2) = 0.$$

   We then find the i values:

   $$i_0 = R^+(100, 2) = 2, i_1 = R^+(49, 2) = 1, i_2 = R^+(24, 2) = 2,$$

   $$i_3 = R^+(11, 2) = 1, i_4 = R^+(5, 2) = 1, i_5 = R^+(2, 2) = 2.$$

   Thus the string is

   $$s_2s_1s_1s_2s_1s_2$$

   (d) To calculate the string in $\{s_1, s_2, s_3, s_4\}$ with number 100, we must first find the u_i values of 100 relative to 4:

   $$u_0 = 100, u_1 = Q^+(100, 4) = 24,$$

   $$u_2 = Q^+(24, 4) = 5, u_3 = Q^+(5, 4) = 1, u_4 = Q^+(1, 4) = 0.$$

   We then find the i values:

   $$i_0 = R^+(100, 4) = 4, i_1 = R^+(24, 4) = 5,$$

   $$i_2 = R^+(5, 4) = 1, , i_3 = R^+(1, 4) = 1.$$

   Thus the string is

   $$s_4s_4s_1s_1$$

3. Let f be a function $\{s_1, \cdots, s_n\}^* \to \{s_1, \cdots s_n\}^*$ which returns s_1 if a string w has a even number of symbols, and 0 otherwise. Write a program in P_n that computes the function f.

   We write a program that begins with $Y = 0$, then alternates back and forth between 1 and 0 as it goes through the string. As it passes through the string, it deletes the characters of the string until nothing is left, then returns that value of Y:

   (A) IF $X \neq 0$ GOTO B
   GOTO E
   (B) IF Y ends s_1 GOTO C
   $Y \leftarrow s_1 Y$
   $X \leftarrow X^-$
   GOTO A
   (C) $Y \leftarrow Y^-$
   $X \leftarrow X^-$
   GOTO A

4. Write a Post-Turing program using that strictly computes the function $s(x) = x+1$ relative to $\{s_1, s_2\}$ (that is, its input is a string $w \in \{s_1 \ldots s_2\}$, and its output is the string in $\{s_1, s_2\}$ corresponding to the number of w + 1).

   The Post-Turing program must perform addition by 1, so it begins at the end of the string, and changes a blank to s_1, an s_1 to an s_2. However, if it reads an s_2, it must read the next character and check again (that it, it carries the 1 until it no longer reads an s_2). Finally, we must also ensure that it ends with the pointer to the left of the data. The program:

   RIGHT TO NEXT BLANK
   (A) LEFT
   IF s_2 GOTO A
   IF s_1 GOTO B
   PRINT s_1
   LEFT TO NEXT BLANK
   GOTO E
   (B) PRINT s_2
   LEFT TO NEXT BLANK

5. If $u \neq 0$, let $n(u, v)$ be the number of occurrences of u as a part of v (for example, $u(ba, ababaa) = 2$); and let $u(0, v) = 0$. Give a Turing machine that strictly computes n.

   Consider the following P_m-program:

   (T) $j \leftarrow i$
   $Z_1 \leftarrow X_1$
   $Z_2 \leftarrow X_2$

```
(L) IF j = 0 GOTO C
j ← j − 1
Z_1 ← Z_1^−
GOTO L
(C) IF Z_2 ends s_i GOTO L_i
(L_0) IF Z_1 ends s_0 GOTO H
(L_i) IF Z_1 ends s_i GOTO A GOTO F
(A) Z_1^−
Z_2^−
GOTO C
(H) Y ← Y + 1
(F) i ← i + 1
IF i + 1 < length(X_1) GOTO T
```

This program calculates n, by succesively looking through parts of the string X_1, and determining if that part is the same as X_2. It then returns the number of those occurences. In class, we saw that any P_m program could be translated into a strict Post-Turing program, and any Post-Turing program can be translated into a strict Turing machine. Thus, we take the above program, apply the two translations, and we get a Turing machine which strictly computes n.