1. The Recursion Theorem

Note. We have been viewing Turing machines as functions from the set of input strings to the resulting condition of the Turing machine, that is $TM : \Sigma^* \rightarrow \{\text{Accept, Reject, Doesn't halt}\}$. We are now considering Turing machines as functions from the input string to the output string that remains on the tape after the Turing machine halts, $TM : \Sigma^* \rightarrow \Sigma^*$. We make this formal with the following definition.

Definition. A function $f : \Sigma^* \rightarrow \Sigma^*$ is a computable function if some Turing machine $TM_f$, on every input $w$, halts with just $f(w)$ on its tape (p. 190).

Thus every computable function results from some Turing machine as the following diagram illustrates.

$$
\begin{array}{c}
TM_f \\
\downarrow \\
f
\end{array}
$$

For any string $w$, define $PRINT_w$ to be a Turing machine that ignores the input, prints out $w$, and then halts. For example, let $g$ be the constant function $\text{const}_{10011}$ so that for all strings $w$, $\text{const}_{10011}(w) = 10011$. Then $TM_g = PRINT_{10011}$, and we have the following diagram.

$$
\begin{array}{c}
PRINT_{10011} \\
\downarrow \\
\text{const}_{10011}
\end{array}
$$

For all computable functions $f$, let $TM_f$ denote the Turing machine that computes the computable function $f$.

Our first goal is to build a Turing machine $SELF$ that ignores the input and produces $\langle SELF \rangle$ as output.

Lemma 1.1. There is a computable function $q : \Sigma^* \rightarrow \Sigma^*$ where $q(w) = \langle PRINT_w \rangle$.

Proof. To show that we can compute $q$, we define $TM_q$ as the following. We are given an input string $w$. Given this input, construct the representation of a Turing machine $PRINT_w$ that erases its input, writes $w$ on the tape, and then halts. Output $\langle PRINT_w \rangle$. \hfill \Box

Lemma 1.2. There is a computable function $p : \Sigma^* \rightarrow \Sigma^*$ such that $p(w) = q(w)w = \langle PRINT_w \rangle w$.

Proof. We construct $TM_p$. Consider a two-tape Turing machine. Copy the input to the second tape, run $TM_q$ on the input on the first tape, then append the original copy of the input to the result. \hfill \Box

Definition. Suppose $M_1$ and $M_2$ are two Turing machines. Define the composition of two Turing machines $M_1 \circ M_2$ to be a new Turing machine that executes $M_1$ until it halts, moves the tape head to the left, then executes $M_2$ on the output of $M_1$. Similarly, suppose $f_1$ and $f_2$ are computable functions. We define the composition $f_1 \circ f_2$ of computable functions in the standard
way: \((f_1 \circ f_2)(x) = f_2(f_1(x))\). Note that \(T_M_{f_1} \circ T_M_{f_2} \equiv T_M_{f_1 \circ f_2}\). We can represent this relation with the following diagram.

\[
\begin{array}{ccc}
\text{Turing machines:} & T_M_{f_1} & \circ & T_M_{f_2} \\
\downarrow & \downarrow & \\
\text{Computable functions:} & f_1 & \circ & f_2
\end{array}
\]

For another example, suppose \(\Sigma = \{1\}\), and let \(d\) be the function that doubles the number of 1’s on the tape, and let \(h\) be the function so that erases half of the 1’s if it’s an even number, otherwise it erases the tape. Then \(T_M_{doh} \equiv \text{DONOTHING}\) because for all strings \(w\), \(h(d(w)) = w\), as is illustrated by this diagram.

\[
\begin{array}{ccc}
\text{Turing machines:} & \text{DONOTHING} & = & T_M_d \circ T_M_h \\
\downarrow & \downarrow & \\
\text{Computable functions:} & \text{identity function} & = & d \circ h
\end{array}
\]

**Theorem.** We can construct a machine \(\text{SELF}\) that prints \(\langle \text{SELF} \rangle\).

**Proof.** Define \(\text{SELF}\) to be \(\text{PRINT}_{\langle T_M_p \rangle} \circ T_M_p\). Then \(\text{SELF} = T_M_{\text{self}}\), where \(\text{self}\) is a computable function. We have the following diagram.

\[
\begin{array}{ccc}
\text{Turing machines:} & \text{SELF} & = & \text{PRINT}_{\langle T_M_p \rangle} \circ T_M_p \\
\downarrow & \downarrow & \\
\text{Computable functions:} & \text{self} & = & \text{const}_{\langle T_M_p \rangle} \circ p
\end{array}
\]

We see that \(\text{self} = \text{const}_{\langle T_M_p \rangle} \circ p\). Given a string \(w\), we compute \(\text{self}(w)\).

\[
\begin{align*}
\text{self}(w) &= \text{p}\left(\text{const}_{\langle T_M_p \rangle}(w)\right) \\
&= \text{p}(\langle T_M_p \rangle) \\
&= \text{q}(\langle T_M_p \rangle) \langle T_M_p \rangle \\
&= \langle \text{PRINT}_{\langle T_M_p \rangle} \rangle \langle T_M_p \rangle.
\end{align*}
\]

We choose a string encoding for Turing machines such that the concatenation of two Turing machine strings represents the concatenation of the two Turing machines, that is \(\langle T_1 \rangle \langle T_2 \rangle = \langle T_1 \circ T_2 \rangle\). Thus

\[
\text{self}(w) = \langle \text{PRINT}_{\langle T_M_p \rangle} \circ T_M_p \rangle = \langle \text{SELF} \rangle,
\]

so \(\text{SELF}\) outputs its own representation. \(\square\)

**Note.** In Sipser’s notation, \(A = \text{PRINT}_{\langle T_M_p \rangle}\) and \(B = T_M_p\).

**Theorem.** Given any computable function \(t : \Sigma^* \times \Sigma^* \rightarrow \Sigma^*\), there is a computable function \(r : \Sigma^* \rightarrow \Sigma^*\), where for every string \(w\),

\[
r(w) = t(\langle T_M_r \rangle, w).
\]

We want such a theorem so we can write computable functions of the form \(t(\text{myself}, w)\) and think of them in the form \(r(w)\) where “myself” is substituted for its string representation as a Turing machine.

**Proof.** First we need to define what we mean by the domain \(\Sigma^* \times \Sigma^*\), the set of pairs of strings. We’ll assume that there’s a character \(\#\) that separates two strings. Next we need to modify our functions \(\text{PRINT}\) and \(p\).

Define \(\text{PRINT}_2\) to compute the function \(\text{print}_x(w) = x^2w\). Now define the function \(p'(x^2w) = \langle \text{PRINT}_2 \rangle x^2w\). This function is computable, and the Turing machine that computes it is similar to \(T_M_p\) in the definition of \(\text{SELF}\).
We are now in a position to construct $TM_r$ as $\text{PRINT}'_{\langle TM_{p'} \circ TM_t \rangle} \circ TM_{p'} \circ TM_t$. Now we verify that this works.

Turing machines: $TM_r = \text{PRINT}'_{\langle TM_{p'} \circ TM_t \rangle} \circ TM_{p'} \circ TM_t$

Computable functions: $r = \text{print}'_{\langle TM_{p'} \circ TM_t \rangle} \circ p' \circ t$

This diagram shows that $r(w) = t \left( p' \left( \text{print}'_{\langle TM_{p'} \circ TM_t \rangle}(w) \right) \right)$. We carry out the evaluation.

\[
\begin{align*}
r(w) &= \quad t \left( p' \left( \text{print}'_{\langle TM_{p'} \circ TM_t \rangle}(w) \right) \right) \\
&= \quad t \left( p' \left( \langle TM_{p'} \circ TM_t \rangle \sharp w \right) \right) \\
&= \quad t \left( \langle \text{PRINT}'_{\langle TM_{p'} \circ TM_t \rangle} \circ TM_{p'} \circ TM_t \rangle \sharp w \right) \\
&= \quad t \left( \langle TM_r \rangle \sharp w \right) \\
&= \quad t \left( \langle TM_r \rangle, w \right)
\end{align*}
\]

as claimed.

\[\square\]

Note. Sipser’s proof of the Recursion Theorem is incorrect. Sipser’s $P_{(BT)}$, destroys the input string as he defines it. It must be modified if his proof is to succeed. His argument works if we set $A = \text{PRINT}'_{\langle TM_{p'} \circ TM_t \rangle}$, $B = TM_{p'}$, $C = TM_t$. 

3