Theorem: \( \mathcal{T} \subseteq \mathcal{R} \).

Proof: Idea is to use coding of Turing machine configurations.

Recall notation from Cutland about primes

- Number primes as \( p_0 = 0, p_1 = 2, p_2 = 3, \ldots \)
- For \( n \in \mathbb{N} \) define \( (n)_y \) = the exponent of \( p_y \) in the prime factorisation of \( x \) for \( x, y > 0 \) and 0 otherwise.

Proofs on page 40-41 that these functions are \textbf{primitive recursive}.

Before we get to details of \( \mathcal{T} \subseteq \mathcal{R} \), let’s talk about primitive recursive codings and sequences a bit.

- Code \( \langle a_0, \ldots, a_{n-1} \rangle \) by
  \[
  2^n 3^{a_0} 5^{a_1} \cdots p_{n+1}^{a_{n-1}}
  \]
  where \( p_0 = 0, p_1 = 2, p_2 = 3, \ldots \). Then,
  \[
  \text{lh}(S) = (S)_1
  \]
  and if we consider \( a_i \) the \((i + 1)\)st entry
  \[
  \text{entry}(S, i) = (S)_{i+2}.
  \]

- OR, code \( \langle a_0, \ldots, a_{n-1} \rangle \) by
  \[
  2^{a_0 + 1} \cdots p_n^{a_{n-1} + 1}.
  \]
  Then length of sequence \( S \) is given by
  \[
  \text{lh}(S) = \mu z < S ((S)_{z+1} = 0)
  \]
  and
  \[
  \text{entry}(S, i) = (S)_{i+1} - 1.
  \]

We will now use these to code all Turing machine operations as numbers and functions on numbers.

Details from Boolos, Burgess, Jeffrey pp. 88-93.

For simplicity, consider unary \( f \in \mathcal{T} \) computed by TM \( M \). We will define general recursive functions that compute encodings of

- Contents of tape
- Configuration of machine
- Transitions between states
- Output when halt.

The general recursive function needs to encapsulate the quadruples of \( M \). Number the states so that 1 is the initial state, 0 is the halting state, and the other states are \( 2, 3, \ldots, \ell \) for some \( \ell \).

Explicitly require there to be quadruples starting \( q, b \) for all \( q \in \{1, \ldots, \ell\} \) and \( b \in \{B, 1\} \). [I.e. add any missing instruction with action ”do nothing” (aka write the symbol that see) and go to halt state 0]
To do so, we will think of the quadruples ordered in some way:

\[\langle 1, B, \dashv, \dashv, \dashv \rangle, \langle 1, 1, \dashv, \dashv, \dashv \rangle, \langle 2, B, \dashv, \dashv, \dashv \rangle, \langle 2, 1, \dashv, \dashv, \dashv \rangle, \ldots, \langle \ell, B, \dashv, \dashv, \dashv \rangle, \langle \ell, 1, \dashv, \dashv, \dashv \rangle.\]

Since we have a tuple for every \(\langle q, b, \dashv, \dashv \rangle\) combination, really just need to keep track of last two elements in quadruple. These can be coded as numbers

- 0: write \(B\)
- 1: write 1
- 2: move L
- 3: move R.
- \(i\): switch to state \(i\) for \(0 \leq i \leq \ell\).

And now we want to encode this as a sequence of numbers

**Example**

The sequence of numbers

\[\langle 1, 0, 2, 1 \rangle\]

Represents the TM with two quadruples

\[\langle 1, B, 1, 0 \rangle, \langle 1, 1, L, 1 \rangle\]

which moves left until first blank and then writes 1 there.

Thus, in string encoding a TM with \(\ell\) many states

- Length is \(4\ell\) (2 quadruples per state, 2 components per quadruple).
- Even numbered entries code actions.
- Odd numbered entries code next states.
- Instruction for action to perform in state \(q\) reading \(b\) is entry

\[4(q - 1) + 2b.\]

- State to transition to after being in state \(q\) reading \(b\) is entry

\[4(q - 1) + 2b + 1.\]

We will use the number which codes the sequence representing all quadruples of \(M\) as a parameter in the general recursive function computing \(f\). But, this function has argument \(x\), which when computed by \(M\) is written out on the tape.

Therefore, we now look at encoding the tape by natural numbers.

We will represent the tape by two numbers (Wang encoding). At any point of computation, the tape is

\[\cdots B B b_0^L \cdots b_0^R b_1^L b_1^R \cdots b_m^R B B \cdots\]

where read/write head is scanning \(b_0^R\). Let

\[n^L = \sum_{i=0}^{n} 2^i b_i^L\]

\[n^R = \sum_{i=0}^{m} 2^i b_i^R.\]
Note that, at the start of computation, all cells to the left of read/write head are blank. Thus, $n^L = 0$. Also,

$$n^R = \sum_{i=0}^{x} 2^i = 2^{x+1} - 1.$$  

We name the function that gives this starting value for the right side of the tape

$$\text{strt}(x) = 2^{x+1} - 1.$$  

Throughout the construction, we will want access to the symbol currently being scanned. This is the least significant bit of $n^R$ so

$$\text{scan}(n^R) = \text{rm}(n^R, 2).$$  

How do $n^L, n^R$ change as the TM runs?

- If action is to write $B$ then

  $$\begin{cases} 
  \text{newleft}_0(n^L, n^R) = \text{wb}^L(n^L, n^R) = n^L \\
  \text{newright}_0(n^L, n^R) = \text{wb}^R(n^L, n^R) = n^R \pm \text{scan}(n^R).
  \end{cases}$$  

- If action is to write 1 then

  $$\begin{cases} 
  \text{newleft}_1(n^L, n^R) = \text{w}1^L(n^L, n^R) = n^L \\
  \text{newright}_1(n^L, n^R) = \text{w}1^R(n^L, n^R) = n^R + 1 \pm \text{scan}(n^R).
  \end{cases}$$  

- If action is to move L then shift $n^L$ to L

  $$\begin{cases} 
  \text{newleft}_2(n^L, n^R) = \text{m}L(n^L, n^R) = \text{qt}(n^L, 2) \\
  \text{newright}_2(n^L, n^R) = \text{m}L(n^L, n^R) = 2 \cdot n^R + \text{rm}(n^L, 2).
  \end{cases}$$  

- If action is to move R then shift $n^R$ to L

  $$\begin{cases} 
  \text{newleft}_3(n^L, n^R) = \text{m}R(n^L, n^R) = 2 \cdot n^L + \text{rm}(n^R, 2) \\
  \text{newright}_3(n^L, n^R) = \text{m}R(n^L, n^R) = \text{qt}(n^R, 2).
  \end{cases}$$  

We combine each of these by cases to get the functions

$$\text{newleft}(n^L, n^R, a) \quad \text{newright}(n^L, n^R, a).$$  

We define one last function directly dealing with contents of the tape: it tells us the value of the output if $M$ has halted in standard configuration. That is, it outputs the number of 1s to the right of the read/write head.

$$\text{val}(n^R) = \lg(n^R, 2) \div 1.$$  

To put the functions we’ve defined based on the contents of the tape together with the coding $m$ of $M$: If tape is defined by $n^L, n^R$ then action required if at state $q$ is

$$\text{actn}(m, q, n^R) = \text{entry}(m, 4(q \div 1) + 2 \cdot \text{scan}(n^R)),$$

The state that will be entered after the action is

$$\text{nxtstate}(m, q, n^R) = \text{entry}(m, 4(q \div 1) + 2 \cdot \text{scan}(n^R) + 1).$$
Goal: describe configuration of $M$ after $t$ steps on input $x$

Configuration = total description of machine snapshot. Can be described by $n^L, n^R, q$ (given parameter $m$).

\[ c(n^L, q, n^R) = 2^{n^L}3^{n^R} \]

Can now define $\text{conf}(m, x, t)$ by primitive recursion.

- Base case: $\text{conf}(m, x, 0) = c(\langle 0, 1, \text{strt}(x) \rangle)$.
- If know configuration at time $t$, can use $m$ to decode action and then result on configuration for time $t + 1$.

\[
\text{conf}(m, x, t + 1) = \text{newconf}(m, \text{conf}(m, x, t)) = \text{newconf}(m, F)
\]

\[
= c(\langle \text{newleft}((F)_1, (F)_3, \text{actn}(m, (F)_2, (F)_3)),
\text{nxtstate}(m, (F)_2, (F)_3),
\text{newright}((F)_1, (F)_3, \text{actn}(m, (F)_2, (F)_3)) \rangle)
\]

Goal: recognise halting

The machine halts when enters state 0, i.e. when

\[(\text{conf}(m, x, t))_2 = 0.\]

Moreover, $M$ is in standard position at halting if and only if in state 0 and have all 1s contiguous, i.e. $r = 2^{\log(r^2)} + 1 = 1$. Hence, halt in standard position if and only if

\[
\text{nstd}(m, x, t) = \begin{cases} 
1 & \text{if } (\text{conf}(m, x, t))_2 \neq 0 \lor (\text{conf}(m, x, t))_3 \neq 2^{\log((\text{conf}(m, x, t))_3) + 1} \neq 1 \\
0 & \text{if } (\text{conf}(m, x, t))_2 = 0 \land (\text{conf}(m, x, t))_3 = 2^{\log((\text{conf}(m, x, t))_3) + 1} \neq 1 
\end{cases}
\]

Therefore, we can recognise halting by

\[
\text{halt}(m, x) = \mu t \ (\text{nstd}(m, x, t) = 0).
\]

Moreover, if $M$ halts in standard position at time $t$ then output of the machine is

\[
\text{out}(m, x, t) = \text{val}( (\text{conf}(m, x, t))_3 )
\]

Putting it all together, we get that

\[
\varphi(x) = F(m, x) = \text{out}(m, x, \text{halt}(m, x)).
\]

We have therefore shown that any TM computable function is general recursive.

Moreover, we have the Theorem: Any general recursive function can be written in standard form as

\[
\varphi(\mu y[\psi(\bar{x}, y)])
\]

where $\varphi, \psi$ are primitive recursive.