# Theory of Computation
# Recursive Function Theory

# Revisiting Integer Computable functions

- Computable functions - computed by Turing Machine algorithms.
- Other direction: Every Turing Machine $M$ can be thought to compute a function $f$ from positive integers to positive integers - where $f$ is a partial recursive function.
- If for input arguments $\{i_1, i_2, \ldots, i_k\}$, the output is $0^m$ followed by blanks, then $f(i_1, i_2, \ldots, i_k) = m$. If the output is otherwise, then $f(i_1, i_2, \ldots, i_k)$ is undefined.
- To be more specific we denote $f$ as $f_M^{(k)}$. If the number of arguments are clear - simply $f_M$.

# Encoding of a TM

- Can we encode a TM to be a valid input of a positive integer computable function? - *integer representation* of a TM by a binary string.
- The first element of the encoding/representation is 1. (We want the encoding of the TM to represent a unique positive integer!)
- Description of a TM $M = (Q, \Sigma, \Gamma, \vdash, \delta, s, t, r)$.
- We can encode each of these elements in unary or of the form $O^{i_1}10^{i_2}\ldots$.
- Eg: $\Sigma$ can be encoded as $0^{|\Sigma|}$. The $j^{th}$ element is encoded as $0^j$.

# Encoding of a Turing Machine cont.

- Eg: $\delta(q_i, a) = (q_j, b, R)$ is encoded as $0^i10^a10^j10^b10^r$, where $0^r$ is the encoding of $R$. Similarly, for $L$.
- Each transition can be separated by $11$. The entire block can be begun with $111$ and ended with $111$ - paddings of $1$ can be allowed as input for a TM computing a function from positive integers to a positive integer; we only look at the maximal blocks of consecutive $0$s.
- This *integer representation* of a TM is the binary representation of a positive integer $i$. Then $i$ is called the *index* of this TM - leading 1 makes the index unique.
- Function $f_M$ can now also be denoted as $f_i$ if $i$ is the index of $M$.

# Functions from integers to TMs

- The integer representation of a TM shows that we can define a function index from the set of all TMs to positive integers: map a TM to its index.
- Suppose I want a function revIndex from positive integers to the set of all TMs:
- If a positive integer has binary representation that is the integer representation of a TM then the integer is mapped to that TM.
- Any other positive integer is mapped to the trivial TM $M_0$ that has a single state (accept state) and on any input string over $\Sigma = \{0, 1\}$ immediately halts and accepts.
- Now a TM can be thought of as an integer and an integer can be thought of as a TM!

# Intermezzo: Rice's Theorem from FLAT

- Essentially these reductions take as input a Turing Machine $M$ (representing a language) and give as output a Turing machine $M'$ (representing another language).
- So consider the index $i$ of $M$ and $j$ of $M'$ as before.
- Such a reduction can be thought of as a total recursive function $g(i) = j$. (Every input TM has a reduced instance)
- A TM computing $g$ does one more step: convert the binary encoding of $M'$ to its unary encoding (Definition of total recursive functions).

# $S_{mn}$ Theorem

Theorem: Let $g(x, y)$ be a partial recursive function on two variables $x, y$ taking positive integer values. There is a total recursive function $\sigma$ on one variable such that $f_{\sigma(x)}(y) = g(x, y)$ for all $x, y$.

Thus, $\sigma(x)$ is the index of a TM $M_x$ such that $f^{(1)}_{M_x}(y) = g(x, y)$ for each $y$.

Need to design a Total TM A that computes the total recursive function $\sigma$.

# $S_{mn}$ Theorem contd.

Proof:

- Let $M$ be a Turing machine computing $g$.
- $A$ is a TM that takes $x$ as input (in unary) and outputs the description of $M_x$, which does the following.
- $M_x$ is a TM that given $y$ (in unary) shifts it to the right and writes $0^x1$ to its left. Then the head is returned to the leftmost position.
- $M_x$ then simulates $M$. (Description of $M_x$ should contain the fact that it simulates $M$, $A$ is not actually simulating $M$!)
- Output of $A$ is the encoding $< M_x >$.
- So $A$ computes the *total recursive* function $\sigma$: On taking in $x$ it *always* outputs $< M_x >$ and $f_{\sigma(x)}(y) = f_{M_x}(y) = g(x, y)$.

# Recursion Theorem

Theorem: For any total recursive function $\sigma$ on one variable taking positive integer values there exists an $x_0$ such that
$f_{x_0}(x) = f_{\sigma(x_0)}(x)$ for all $x$.
Think of $\sigma$ as mapping indices of TMs (partial recursive functions) into indices of TMs (partial recursive functions) - and they have a fixed point.
Fixed point of $\sigma$: Original TM function is same as modified TM function.

# Recursive Theorem contd.

Proof:

- Consider you have access to a Universal TM that has a special tape, where it is enumerating all TMs (writing down descriptions of TMs one after another).
- Starting from 0 and going upwards, it writes down the binary string of a positive integer.
- It checks if this can be a valid integer representation of a TM $M$.
- If not, then on the special tape it writes down the description of trivial TM $M_0$.
- Valid integer representation - it writes down the integer representation of $M$ onto its special tape. So the index of a non-trivial $M$ can also be thought of as the index of enumeration of $M$ by the UTM.

# Recursive Theorem contd.

Proof: Coming back to showing a fixed point for $\sigma$,

- For each positive integer $i$ construct a TM $M_i$ that takes in $x$, computes $f_i(i)$ and simulates the $f_i(i)$th TM, generated by the UTM, on $x$. If this TM is non-trivial, then its index is also $f_i(i)$.
- Let $M_i$ be the TM with index $g(i)$. For all $i, x$, $f_{g(i)}(x) = f_{f_i(i)}(x)$.
- $g(i)$ is a total recursive function (even if $f_i(i)$ is not defined) - only needs the description of the UTM and instructions to find out the $f_i(i)$th TM.

# Recursive Theorem contd.

Proof contd.:

- Consider the composite function $\sigma g$. It is also total recursive. Let $j$ be the index of a TM computing $\sigma g$, so $\sigma g = f_j$.
- Take $x_0 = g(j)$.
- $f_{x_0}(x) = f_{g(j)}(x) = f_{f_j(j)}(x)$
  $= f_{\sigma(g(j))(x)}$ $(f_j = \sigma g)$.
  $= f_{\sigma(x_0)}(x)$.
- So TM with index $x_0$ and TM with index $\sigma(x_0)$ compute the same function.

# Application of the theorems

Let $M_1, M_2, \ldots$ be any enumeration of all Turing machines. We show that for some $i$, $M_i$ and $M_{i+1}$ both compute the same function.

# Application of the theorems contd.

- Define total recursive function $\sigma(i)$ as follows:
- (i) Enumerate TMs $M_1, M_2, \ldots$ until the one which has index $i$.
- (ii) Let TM $M_j$ have index $i$. Enumerate one more TM $M_{j+1}$ and $\sigma(i)$ is set to be the index for $M_{j+1}$.
- Recursion theorem: There is some $x_0$ where $M_{x_0}$ and $M_{x_0+1}$ define the same function of one variable.

# Other Problems

- Prove that there exists $x_0 \in N$ such that for all $y$, $f_{x_0}(y)$ is $y^2$ if $y$ is even, and $f_{x_0+1}(y)$ otherwise.
- Define any fixed point for the total recursive function $\sigma : N \to N$ defined as follows: for $x \in N$, the TM with description $\sigma(x)$ computes the function $f_{\sigma(x)}(y)$ which is $1$ if $y = 0$ and $f_x(y+1)$ otherwise.
  Describe a fixed point for $\sigma$.
- Let $\sigma : N \to N$ be any total recursive function. Prove that $\sigma$ has infinitely many fixed points i.e., there are infinitely many $w \in N$ such that $f_w(y) = f_{\sigma(w)}(y)$ for all $y$.