The Recursion Theorem Concept

- **Let $g(z, x_1, \ldots, x_m)$ be a partially computable function of $m+1$ variables. Then, there is a number e such that**

$$\Phi^{(m)}(x_1, \ldots, x_m, e) = g(e, x_1, \ldots, x_m)$$

- **Surprising result:** A $m+1$ variables partially computable function g can be computed by converting one of its input variables to a constant, which becomes the program that computes g!
- **What does this mean?**
  1. The program that computes g has a program number e.
  2. Universal function $\Phi^{(m)}()$ only takes x_1, ..., x_m as input for program e and delivers an output identical to $g(e, x_1, \ldots, x_m)$.
  3. Since its program, e, is an input variable, **can we compute this e in g?**
  4. The answer is **no!** because it is impossible.
  5. However, we can generate e *implicitly*. Here is how.

- Consider the following program $\mathcal{Q}$ and $q = \#(\mathcal{Q})$

$$\begin{aligned} Z &\leftarrow S_m^1(X_{m+1}, X_{m+1}) \\ Y &\leftarrow g(Z, X_1, \ldots, X_m) \end{aligned}$$

  This program takes X_{m+1} as a program and runs it on input X_{m+1}. The output is given to function g.

- We want to "generate" the value of q. The easiest way is inserting $X_{m+1} \leftarrow X_{m+1} + 1$ q times so that the value of X_{m+1} is q. Let this program be $\mathcal{R}$:

$$\begin{aligned} X_{m+1} &\leftarrow X_{m+1} + 1 \\ X_{m+1} &\leftarrow X_{m+1} + 1 \\ &\vdots \\ X_{m+1} &\leftarrow X_{m+1} + 1 \\ Z &\leftarrow S_m^1(X_{m+1}, X_{m+1}) \\ Y &\leftarrow g(Z, X_1, \ldots, X_m) \end{aligned}$$

- What does $\mathcal{R}$ compute?

  1. Function $S_m^1(X_{m+1}, X_{m+1})$ is actually called with $S_m^1(q, q)$.
  2. From the $s - m - n$ theorem, $S_m^1(q, q)$ will insert q copies of $X_{m+1} \leftarrow X_{m+1} + 1$ and then compute q (*i.e.*, program $\mathcal{Q}$)

$$
\begin{aligned}
X_{m+1} &\leftarrow X_{m+1} + 1 \\
X_{m+1} &\leftarrow X_{m+1} + 1 \\
&\vdots \\
X_{m+1} &\leftarrow X_{m+1} + 1 \\
Z &\leftarrow S_m^1(X_{m+1}, X_{m+1}) \\
Y &\leftarrow g(Z, X_1, \ldots, X_m)
\end{aligned}
$$

3. **But, what $S_m^1(q, q)$ computes is actually program $\mathcal{R}$ which has q copies of $X_{m+1} \leftarrow X_{m+1} + 1$ followed by program $\mathcal{Q}$.**
4. Therefore, in program $\mathcal{R}$, function g computes $g(\#(\mathcal{R}), x_, \ldots, x_m)$ and the universal function is

   $$\Phi^{(m)}(x_1, \ldots, x_m, \#(\mathcal{R})) = g(\#(\mathcal{R}), x_1, \ldots, x_m)$$

   And we are done!

The Recursion Theorem
A Formal Proof

- Consider the following partially computable function:

$$g(S_m^1(v, v), x_1, \ldots, x_m)$$

  where v is an arbitrary number.

- Since the above function is partially computable, we can find a program to compute it. Let this program's number be p. Therefore, by the $s - m - n$ theorem, we have:

$$\begin{aligned} g(S_m^1(v, v), x_1, \ldots, x_m) &= \Phi^{(m+1)}(x_1, \ldots, x_m, v, p) \\ &= \Phi^{(m)}(x_1, \ldots, x_m, S_m^1(v, p)) \end{aligned}$$

- Since v is arbitrarily chosen, we can use $v = p$:

$$g(S_m^1(p, p), x_1, \ldots, x_m) = \Phi^{(m)}(x_1, \ldots, x_m, S_m^1(p, p))$$

- Letting $e = S_m^1(p, p)$, we have the desired result:

$$g(e, x_1, \ldots, x_m) = \Phi^{(m)}(x_1, \ldots, x_m, e)$$

The Halting Problem – Take Four
Proved by the Recursion Theorem

- Suppose $\mathrm{HALT}(x, y)$ were computable. Consider the following function f:

$$f(x,y) = \begin{cases} \uparrow & \text{if } \mathrm{HALT}(y,x) \\ \\ 0 & \text{otherwise} \end{cases}$$

  Therefore, $f(x,y)\uparrow$ iff program x halts on input y.

- By the Recursion Theorem, there exists a e such that

$$\Phi(y,e) = f(e,y) = \begin{cases} \uparrow & \text{if } \mathrm{HALT}(y,e) \\ \\ 0 & \text{otherwise} \end{cases}$$

- Therefore, we have

$$\Phi(y,e)\uparrow \Leftrightarrow \mathrm{HALT}(y,e)$$

- $\Phi(y,e)\uparrow$ is equivalent to "program e does not halt on input y," which is exactly $\neg\mathrm{HALT}(y,e)$.

- Hence, we have

$$\neg\mathrm{HALT}(y,e) \Leftrightarrow \mathrm{HALT}(y,e)$$

  This is a contradiction and $\mathrm{HALT}(x,y)$ cannot be computable.

The Recursion Theorem
Some Immediate Results

- **There is a number e such that $\Phi(x, e) = e$ holds for all x.**

  1. It means there is a program such that *when it is run on any input the output is "itself."*
  2. **Proof:** Let $g(z, x) = \pi_1^2(z, x) = z$. By the Recursion Theorem, there is a e such that $\Phi(x, e) = g(e, x) = \pi_1^2(e, z) = e$.

- **Fixed Point Theorem: Let f be a computable function. Then, there is a number e such that $\Phi(x, e) = \Phi(x, f(e))$.**

  Proof: Let $g(z, x) = \Phi(x, f(z))$. That is, $g(z, x)$ is the universal function of running program $f(z)$ on input x. By the Recursion Theorem, there exists a e such that

$$\Phi(x, e) = g(e, x) = \Phi(x, f(e))$$

- **If f is a computable function, there is a e such that $W_e = W_{f(e)}$.**

  Proof: By the Fixed Point Theorem, there is a e such that $\Phi(x, e) = \Phi(x, f(e))$. Since $W_e = \{n | \Phi(n, e)\downarrow\}$, $W_{f(e)} = \{n | \Phi(n, f(e))\downarrow\}$ and $\Phi(x, e) = \Phi(x, f(e))$, we have $W_e = W_{f(e)}$.

- **There exists a number e such that $\Phi(x, e) = x^e$ for all x.**

  Proof: Let $g(z, x) = x^z$. Since g is computable, there is a e such that $\Phi(x, e) = g(e, x) = x^e$.

- If function f is a fast increasing function (*i.e.*, $f(z) = 2^{2^{2^{\cdots^{2^z}}}}$), $f(e)$ can be made very large.

- **If f is a computable function, there are arbitrarily large numbers e such that $\Phi(x, e) = \Phi(x, f(e))$.**

  Proof: This is an extension to the Fixed Point Theorem. Take an arbitrarily number n. Then, consider the program sequence $\Phi_0()$, $\Phi_1()$, ..., $\Phi_n()$. Now choose a number c such that $\Phi_c()$ is not one of the above. Since there are infinitely many programs, this c exists. Consider function g:

$$g(y) = \begin{cases} c & y \leq n \\ \\ f(y) & y > n \end{cases}$$

  Since g is computable, by the Fixed Point Theorem, there is a number e such that $\Phi(x, e) = \Phi(x, g(e))$.

  1. We claim that $e > n$ must hold. If $e \leq n$, then $g(e) = c$ and $\Phi_e() = \Phi_{g(e)}() = \Phi_c()$. This is a contradiction because $\Phi_c()$ is different from all $\Phi_0()$, $\Phi_1()$, ..., $\Phi_n()$.
  2. Since $e > n$, $g(e) = f(e)$ and $\Phi_e() = \Phi_{g(e)}() = \Phi_{f(e)}()$ (*i.e.*, $\Phi(x, e) = \Phi(x, f(e))$, and e is also a fixed point of f. Since e is larger than any arbitrarily chosen number n, the proposition is proved.

  Since n is an arbitrary number, there are an infinite number of such e's.

Extended Recursion Theorem

- **Let $g_i(z_1, z_2, \ldots, z_n, x)$, $1 \leq i \leq n$, be n partially computable functions. Then, there exists e_1, e_2, ..., e_n such that**

$$\begin{aligned}
\Phi(x, e_1) &= g_1(e_1, e_2, \ldots, e_n, x) \\
\Phi(x, e_2) &= g_2(e_1, e_2, \ldots, e_n, x) \\
&\vdots \\
\Phi(x, e_n) &= g_n(e_1, e_2, \ldots, e_n, x)
\end{aligned}$$

  Note that x can be replaced with $x_1, x_2, \ldots, x_m$ without affecting the proof below.

- This extends the original version from one "program" (*i.e.*, e) to multiple "programs" (*i.e.*, e_1, e_2, ..., e_n) acting together.

- For $1 \leq i \leq n$, consider function

$$S^n_{(i)1}(z_1, z_2, \ldots, z_n, z_i)$$

Function $S^n_{(i)1}(z_1, \ldots, z_n, z_i)$ runs z_i on n parameters z_1, z_2, ..., z_n.

- Substituting $S^n_{(i)1}()$ for z_i in $g_i()$, we have

$$g_i(S^n_{(1)1}(z_1, \ldots, z_n, z_1), \ldots, \ S^n_{(n)1}(z_1, \ldots, z_n, z_n), x)$$

- Since g_i is partially computable, there is a program p_i that computes g_i:

$$g_i(S^n_{(1)1}(z_1, \ \ldots \ , z_n, z_1), \ldots, \ S^n_{(n)1}(z_1, \ldots, z_n, z_n), x)$$
$$= \ \Phi(x, z_1, \ldots, z_n, p_i)$$

- Define e_1, ..., e_n as follows:

$$e_i = S^n_{(i)1}(p_1, p_2, \ldots, p_n, p_i)$$

- We shall show that e_i's are what we want. For $1 \leq i \leq n$, we have

$\Phi(x, e_i)$
$= \ \Phi(x, S^n_{(i)1}(p_1, p_2, \ldots, p_n, p_i))$
$= \ \Phi(x, p_1, p_2, \ldots, p_n, p_i)$ [the s-m-n Theorem]
$= \ g_i(S^n_{(1)1}(p_1, \ldots, p_n, p_1), \ldots, S^n_{(n)1}(p_1, \ldots, p_n, p_n), x)$
[the definition of g_i]
$= \ g_i(e_1, e_2, \ldots, e_n, x)$ [the definition of e_i]

- Thus, the extended Recursion Theorem is established.

Rice's Theorem: I
Fundamentals

- Consider the universal function that interprets program $\mathcal{P}$ whose number is $p = \#(\mathcal{P})$. Since we focus on the function itself rather than its input, this universal function is written as $\Phi_p^{(1)}$. Since the number of arguments is *always* 1, the superscript is dropped.
- Consider a collection Γ of programs Φ_t.
- Let set R_Γ be defined as follows:

$$R_\Gamma = \{t | \Phi_t \in \Gamma\}$$

  Set R_Γ is called the **index set**.
- An index set must satisfy the following condition:

$$\textbf{If } t \in R_\Gamma \textbf{ and } \Phi_t = \Phi_q \textbf{ then } q \in R_\Gamma$$

- **Meaning?**
  1. R_Γ is recursive $\Leftrightarrow$ $\Phi_t \in \Gamma$ is computable.
  2. Testing if "a program is in Γ" is computable reduces to "proving R_Γ is recursive."

3. Take Γ to be the set of all total functions. Then, "testing if a function f being total (*i.e.*, $f \in \Gamma$) is computable" is equivalent to say that the index set R_Γ is recursive.
4. If index set R_Γ is recursive, we can find a computable predicate P such that

$$x \in R_\Gamma \Leftrightarrow P(x) \text{ is TRUE}$$

5. By the definition of index set R_Γ, the above becomes

$$x \in R_\Gamma \Leftrightarrow \Phi_x \in \Gamma \Leftrightarrow P(x) \text{ is TRUE}$$

   In other word, R_Γ being recursive provides us a way of testing if a function is of certain type.
6. **Meaning:**
   If R_Γ is recursive, then there exists a predicate P which accepts a program x and returns TRUE or FALSE depending on if Φ_x does or does not belong to Γ.

Rice's Theorem: II
Not Every Set Is an Index Set

- Let $\Gamma = \{\Phi_t() | \Phi_t() = \Phi_{t+1}()\}$. Then, R_Γ is **not** an index set. We shall prove it by contradiction. Assume that R_Γ is an index set.
- Consider the successor function $s(x) = x + 1$. By the Fixed Point theorem, there exists a number e such that $\Phi(x, e) = \Phi(x, s(e)) = \Phi(x, e + 1)$. Therefore, we have $\Phi_e = \Phi_{e+1}$. Since $\Phi_e = \Phi_{e+1}$, $e \in R_\Gamma$.
- Recall that an index set satisfies

  $$\textbf{If } k \in R_\Gamma \textbf{ and } \Phi_k = \Phi_j, \textbf{then } j \in R_\Gamma$$

  Since R_Γ is an index set by our assumption and since $e \in R_\Gamma$ and $\Phi_e = \Phi_{e+1}$, we have $e + 1 \in R_\Gamma$.
- Since $e + 1 \in R_\Gamma$, $\Phi_{e+1} \in \Gamma$, which implies $\Phi_{e+1} = \Phi_{(e+1)+1}$. Applying the same argument, we have $e + 2 = (e + 1) + 1 \in R_\Gamma$. By induction, we have

  $$\text{for } i \geq 0, \Phi_{e+i} = \Phi_e \text{ and } e + i \in R_\Gamma$$

- Therefore, we have at most $e+1$ **distinctive** partially computable programs Φ_0, Φ_1, ..., Φ_{e-1}, $\Phi_e = \Phi_{e+1} = \Phi_{e+2} = \cdots$. This is a contradiction because there are infinitely many partially computable programs.

Rice's Theorem: III
The Proposition

- **Let Γ be a collection of partially computable functions of one variable.**

$$R_\Gamma \textbf{ is recursive} \Leftrightarrow R_\Gamma = \emptyset \textbf{ or } R_\Gamma = \mathcal{N}$$

- **Proof:** If $R_\Gamma = \emptyset$ or $R_\Gamma = \mathcal{N}$, R_Γ is certainly recursive. Therefore, we shall prove that if R_Γ is recursive, then $R_\Gamma = \emptyset$ or $R_\Gamma = \mathcal{N}$. We shall prove it by contradiction.

  1. Suppose R_Γ is recursive and R_Γ is neither $\emptyset$ nor $\mathcal{N}$.
  2. There exists a $f \in R_\Gamma$ because $R_\Gamma \neq \emptyset$ and a $g \notin R_\Gamma$ because $R_\Gamma \neq \mathcal{N}$.

3. Consider the following function h:

$$h(y,x) = \begin{cases} \Phi_f(x) & y \notin R_\Gamma \\ \\ \Phi_g(x) & y \in R_\Gamma \end{cases}$$

Since we assume R_Γ is recursive, h is a partially computable function. By the Recursion Theorem, there exists a number e such that $\Phi_e(x) = \Phi^{(1)}(x,e) = h(e,x)$.

4. Now we have

$$\begin{aligned} e \in R_\Gamma &\Rightarrow \Phi_e \in \Gamma \\ &\Rightarrow \Phi_e(x) = h(e,x) = \Phi_g(x) \\ &\Rightarrow g \in R_\Gamma \qquad (\Rightarrow\Leftarrow) \end{aligned}$$

and

$$\begin{aligned} e \notin R_\Gamma &\Rightarrow \Phi_e \notin \Gamma \\ &\Rightarrow \Phi_e(x) = h(e,x) = \Phi_f(x) \\ &\Rightarrow f \notin R_\Gamma \qquad (\Rightarrow\Leftarrow) \end{aligned}$$

5. Therefore, we have a contradiction. As a result, R_Γ cannot be recursive.

Three Undecidable Problems

- **Let Γ_1 be the set of all computable functions, Γ_2 be the set of all primitive recursive functions, and Γ_3 be the set of all partially computable functions that are defined for all but a finite number of values of x.**
- **Then, R_{Γ_1}, R_{Γ_2} and R_{Γ_3} are not recursive.**
- In other words, there are no programs for testing a given program $\mathcal{P}$ to determine whether $\Psi_{\mathcal{P}}^{(1)}(x)$ belongs to one of the three sets.
- **Proof:** Consider the following two functions:

  $$\begin{aligned} f(x) &= s(x) && \text{the successor function} \\ g(x) &= 1 - x && \text{only defined for 0 and 1} \end{aligned}$$

  Let the program numbers of f and g be $\#f$ and $\#g$. Then, $\Phi_{\#f} \in \Gamma_i$ and $\Phi_{\#g} \notin \Gamma_i$, and $\#f \in R_{\Gamma_i}$ and $\#g \notin R_{\Gamma_i}$ for $i = 1, 2, 3$.
- Please verify that R_{Γ_i} is an index set. By Rice's theorem, R_{Γ_i} is not recursive.

Two More Undecidable Problems

- **Testing if a function is a zero function is not computable.**

  Proof: Define Γ and functions f and g as follows:

  $$\begin{aligned} \Gamma &= \text{all functions whose output is always zero} \\ f(x) &= n(x) \quad \text{the zero function} \\ g(x) &= s(x) \quad \text{the successor function} \end{aligned}$$

  Then, $f \in \Gamma$ and $g \notin \Gamma$. By Rice's theorem, R_Γ is not recursive. *Please fill in the details.*

- This proposition can easily be extended to testing for constant functions (*i.e.*, $f(x) = k$ for all x, where k is a constant)

- **Testing if the output from two arbitrary functions are equal is not computable.**

  Proof: If this were recursive, testing if a function is a constant function would also be recursive. This is a contradiction.

Rice's Theorem: Extension I

- **Suppose R_Γ is an index set. Moreover, suppose there is a partially computable function θ such that**

  1. $\{x|\Phi_x = \theta\} \subseteq R_\Gamma$

  2. there is a partially computable function $\hat{\theta}$ such that $\hat{\theta}$ is an *extension* of θ (*i.e.*, $\theta \subseteq \hat{\theta}$) and $\{x|\Phi_x = \hat{\theta}\} \subseteq \bar{R}_\Gamma$

  Then, the index set R_Γ is not r.e.

- Function f is an extension of g (*i.e.*, $g \subseteq f$) if $g(x)$ is defined then $f(x)$ is also defined and $f(x) = g(x)$. More precisely, Domain$(g) \subseteq$ Domain(f) and f and g are identical on the common domain.

- Note that condition (2) above forces $\theta \neq \hat{\theta}$ and therefore θ cannot be total. (**Why?**)

- **Meaning**: if we can find a function, whose index is in R_Γ, such that its extension's index is not in R_Γ, then R_Γ is not r.e. **The set of programs satisfying a given non-total input-output behavior cannot be r.e.**

- We shall prove $\bar{K} \leq_{\mathrm{m}} R_\Gamma$.

- Consider the following function:

$$g(y,x) = \begin{cases} \hat{\theta}(y) & x \in K \\ \\ \theta(y) & \text{otherwise} \end{cases}$$

  $g(y,x)$ is partially computable. **(why?)**

- By the s-m-n Theorem, we have a $f(x)$ such that

  $g(y,x) = \Phi(y,x,\#g) = \Phi(y, S_1^1(x,\#g)) = \Phi(y,f(x))$

- Now we prove $x \in K \Rightarrow f(x) \in \bar{R}_\Gamma$:

$$\begin{aligned} x \in K &\Rightarrow (\forall y)[g(y,x) = \hat{\theta}(y)] \\ &\Rightarrow (\forall y)[\Phi(y,f(x)) = \hat{\theta}(y)] \\ &\Rightarrow \Phi_{f(x)} = \hat{\theta} \\ &\Rightarrow f(x) \in \bar{R}_\Gamma \Rightarrow f(x) \notin R_\Gamma \end{aligned}$$

- Then, we prove $x \notin K \Rightarrow f(x) \in R_\Gamma$:

$$\begin{aligned} x \notin K &\Rightarrow (\forall y)[g(y,x) = \theta(y)] \\ &\Rightarrow (\forall y)[\Phi(y,f(x)) = \theta(y)] \\ &\Rightarrow \Phi_{f(x)} = \theta \\ &\Rightarrow f(x) \in R_\Gamma \end{aligned}$$

- Therefore, $x \notin K \Leftrightarrow f(x) \in R_\Gamma$ and $\bar{K} \leq_{\mathrm{m}} R_\Gamma$.

Rice's Theorem: Extension I Applications

- $R_\Gamma = \{\ x \mid \Phi_x$ is one-to-one $\}$ is not r.e.
  1. Let $\theta(1) = 1$, $\theta(2) = 2$, and undefined elsewhere. Then, $\{x|\Phi_x = \theta\} \subseteq R_\Gamma$.
  2. Let $\hat{\theta}(1) = 1$, $\hat{\theta}(2) = 2$, $\hat{\theta}(3) = 2$, and undefined elsewhere. Then, $\hat{\theta}$ is an extension of θ; but $\hat{\theta}$ is *not* one-to-one. Hence, $\{x|\Phi_x = \hat{\theta}\} \subseteq \bar{R}_\Gamma$.
  3. Therefore, R_Γ is not r.e.
- $R_\Gamma = \{\ x \mid$ Domain(Φ_x) is finite $\}$ is not r.e.
  1. Let $\theta(1) = 1$ and undefined elsewhere. Then, $\{x|\Phi_x = \theta\} \subseteq R_\Gamma$.
  2. Let $\hat{\theta}(x)$ be the identity function. Then, $\hat{\theta}$ is an extension of θ; but, the domain of θ is not finite. Hence, $\{x|\Phi_x = \hat{\theta}\} \subseteq \bar{R}_\Gamma$.
  3. Therefore, R_Γ is not r.e.
- Can you use this extension to prove $\{x|\Phi_x$ is total$\}$ is not r.e.?
  1. No. **Why?**
  2. Thus, this extension cannot cover all non-r.e. sets.
  3. Show that $R_\Gamma = \{x|\Phi_x$ is not total$\}$ is not r.e.

Rice's Theorem: Extension II
Rice-Shapiro Theorem

- **Suppose R_Γ is an index set. Moreover, suppose there is a partially computable function θ such that**

  1. $\{x | \Phi_x = \theta\} \subseteq R_\Gamma$

  2. $\{\ x \mid \Phi_x \subseteq \theta$ **and Domain**(Φ_x) **is finite** $\} \subseteq \bar{R}_\Gamma$

  Then, the index set R_Γ is not r.e.

- The second condition states that **every program that computes a finite portion of θ is not in** R_Γ.

- **Meaning**: A property associated with some partially computable computation but not with its any finite restriction cannot be r.e.

- We shall prove $\bar{K} \leq_{\mathrm{m}} R_\Gamma$

- Consider the following function:

$$g(y,x) = \begin{cases} \theta(y) & \text{if } \neg\text{STEP}^1(x,\#x,y) \\ \\ \uparrow & \text{otherwise} \end{cases}$$

  $g(y,x)$ is a partially computable. (**why?**)

- By the s-m-n Theorem, we have a $f(x)$ such that

  $g(y,x) = \Phi(y,x,\#g) = \Phi(y, S_1^1(x,\#g)) = \Phi(y, f(x))$

- Now, we prove $x \in K \Rightarrow f(x) \in \bar{R}_\Gamma$:

$$\begin{aligned} x \in K &\Rightarrow \Phi(x,x)\downarrow \\ &\Rightarrow (\exists t)[\Phi_x \text{ stops on } x \text{ in exactly } t \text{ steps}] \end{aligned}$$

  1. If $y < t$, $\text{STEP}^1(x,\#x,y)$ is FALSE, because x takes t steps to stop. Thus, $\Phi(y,f(x)) = \theta(y)$ for $y < t$. More precisely, for $y < t$, $\Phi_{f(x)} = \theta$.
  2. If $y \geq t$, $\text{STEP}^1(x,\#x,y)$ is TRUE because in $t \leq y$ steps x stops. In this case, $\Phi(y,f(x))\uparrow$.
  3. In summary, $\Phi_{f(x)} \subseteq \theta$ and $\text{Domain}(\Phi_{f(x)})$ is $\{0,1,\ldots,t-1\}$ which is finite. Therefore, $f(x) \in \bar{R}_\Gamma$

- Thus, we have completed the proof of

$$x \in K \Rightarrow f(x) \in \bar{R}_\Gamma$$

- Recall the definition of $g(y, x)$:

$$g(y,x) = \begin{cases} \theta(y) & \text{if } \neg\text{STEP}^1(x, \#x, y) \\ \\ \uparrow & \text{otherwise} \end{cases}$$

- We shall prove $x \notin K \Rightarrow f(x) \in R_\Gamma$:

$$\begin{aligned} x \notin K &\Rightarrow \Phi(x,x)\uparrow \\ &\Rightarrow (\forall y)[\neg\text{STEP}^1(x, \#x, y)] \\ &\Rightarrow (\forall y)[g(y,x) = \theta(y)] \\ &\Rightarrow (\forall y)[\Phi(y, f(x)) = \theta(y)] \\ &\Rightarrow \Phi_{f(x)} = \theta \\ &\Rightarrow f(x) \in R_\Gamma \end{aligned}$$

- Therefore, we have $x \notin K \Leftrightarrow f(x) \in R_\Gamma$ and hence $\bar{K} \leq_{\text{m}} R_\Gamma$.

Rice-Shapiro Theorem Applications

- $R_\Gamma = \{\ x \mid \Phi_x \text{ is total } \}$ is not r.e.

  1. Let θ be the identity function. Since it is total, $\theta \in R_\Gamma$.
  2. Every restriction f to θ with a finite domain cannot be total. Thus, $f \in \bar{R}_\Gamma$
  3. Therefore, $R_\Gamma =$ TOTAL cannot be r.e.

  Thus, testing if a function is total is not r.e.

- $R_\Gamma = \{\ x \mid \Phi_x = f, f \text{ is computable } \}$ is not r.e.

  1. Let θ be f. Since f is computable, there is a program g that computes f (*i.e.*, $\Phi_g(x) = f(x)$). Then, $g \in R_\Gamma$.
  2. Every restriction of f to a finite domain cannot be a member of R_Γ because it is not total.
  3. Therefore, R_Γ is not r.e.

  Thus, testing if a program has identical input/output activities to a given computable function is not r.e.

Rice's Theorem: Extension III

- Set $A \subseteq \mathcal{N}^n$ is **recursive** if there is a computable predicate $P(x_1, x_2, \ldots, x_n)$ such that

  $$A = \{ (x_1, x_2, \ldots, x_n) \mid P(x_1, x_2, \ldots, x_n) \text{ is TRUE} \}$$

- Set $A \subseteq \mathcal{N}^n$ is **r.e.** if there is a partially computable function $f(x_1, x_2, \ldots, x_n)$ such that

  $$A = \{ (x_1, x_2, \ldots, x_n) \mid f(x_1, x_2, \ldots, x_n)\downarrow \}$$

- A set $R_\Gamma \subseteq \mathcal{N}^n$ is an index set if $(a_1, a_2, \ldots, a_n) \in R_\Gamma$ and $\Phi_{a_1} = \Phi_{b_1}$, $\Phi_{a_2} = \Phi_{b_2}$, ..., $\Phi_{a_n} = \Phi_{b_n}$, then $(b_1, b_2, \ldots, b_n) \in R_\Gamma$.

Suppose $R_\Gamma \subseteq \mathcal{N}^n$ is an index set. Then, R_Γ is recursive if and only if $R_\Gamma = \emptyset$ or $R_\Gamma = \mathcal{N}^n$.

- Since $\emptyset$ and $\mathcal{N}^n$ are r.e., we shall assume R_Γ is recursive and $R_\Gamma \neq \emptyset$ and $R_\Gamma \neq \mathcal{N}^n$.
- Since $R_\Gamma \neq \emptyset$, there exists $(f_1, f_2, \ldots, f_n) \in R_\Gamma$.
- Since $R_\Gamma \neq \mathcal{N}^n$, there exists $(g_1, g_2, \ldots, g_n) \notin R_\Gamma$.
- For $1 \leq i \leq n$, construct function h_i as follows:

$$h_i(y_1, y_2, \ldots, y_n, x) = \begin{cases} \Phi_{f_i}(x) & (y_1, y_2, \ldots, y_n) \notin R_\Gamma \\ \\ \Phi_{g_i}(x) & (y_1, y_2, \ldots, y_n) \in R_\Gamma \end{cases}$$

  Since R_Γ is recursive, both $(y_1, y_2, \ldots, y_n) \notin R_\Gamma$ and $(y_1, y_2, \ldots, y_n) \in R_\Gamma$ are computable. As a result, for every $1 \leq i \leq n$, h_i is partially computable.
- By the extended Recursion Theorem, there are e_1, e_2, ..., e_n such that

$$\Phi_{e_i}(x) = \Phi(x, e_i) = h_i(e_1, e_2, \ldots, e_n, x) \qquad \text{for all } i$$

- If $(e_1, e_2, \ldots, e_n) \in R_\Gamma$, we have

$$\begin{aligned} (e_1, e_2, \ldots, e_n) \in R_\Gamma &\Rightarrow (\forall i)[\Phi_{e_i}(x) = h_i(\ldots, x) = \Phi_{g_i}(x)] \\ &\Rightarrow (g_1, g_2, \ldots, g_n) \in R_\Gamma \quad (\Rightarrow\Leftarrow) \end{aligned}$$

- If $(e_1, e_2, \ldots, e_n) \notin R_\Gamma$, we have

$$\begin{aligned} (e_1, e_2, \ldots, e_n) \notin R_\Gamma &\Rightarrow (\forall i)[\Phi_{e_i}(x) = h_i(\ldots, x) = \Phi_{f_i}(x)] \\ &\Rightarrow (f_1, f_2, \ldots, f_n) \notin R_\Gamma \quad (\Rightarrow\Leftarrow) \end{aligned}$$

Rice's Theorem: Extension III Applications

- $R_\Gamma = \{ (x, y) \mid \Phi_x = \Phi_y \}$ is not recursive.

  1. Note that R_Γ is an index set.
  2. Let $f(x) = x$ and $g(x) = x^2/x$. Let p and q be the program numbers such that $\Phi_p = f$ and $\Phi_q = g$. Thus, $(p, q) \in R_\Gamma$ and R_Γ is not empty.
  3. Let $f(x) = x$ and $g(x) = x^2$. Let p and q be the program numbers such that $\Phi_p = f$ and $\Phi_q = g$. Then, since $\Phi_p \neq \Phi_q$, $(p, q) \notin R_\Gamma$ and $R_\Gamma \neq \mathcal{N}^2$.
  4. Therefore, R_Γ is not recursive.

  Thus, testing if two functions, taking the same input, will produce the same output is not computable.

- $R_\Gamma = \{ (x, y, z) \mid \Phi_z = \Phi_y \circ \Phi_x \}$ is not recursive.

  Thus, testing if a function is the composition of two other functions is not computable.