Recursion Theorems*

Xiaofeng Gao

Department of Computer Science and Engineering
Shanghai Jiao Tong University, P.R.China

CS363-Computability Theory

* Special thanks is given to Prof. Yuxi Fu for sharing his teaching materials.

Outline

1. First Recursion Theorem
   - Recursive Operator
   - Second Order Computable Function
   - First Recursion Theorem
2. Second Recursion Theorem
   - Second Recursion Theorem
   - The Diagonal Argument

Outline

Operator

Let us denote by $\mathscr{F}_n$ the class of all partial functions from $\mathbb{N}^n$ to $\mathbb{N}$.

Operator

Let us denote by $\mathscr{F}_n$ the class of all partial functions from $\mathbb{N}^n$ to $\mathbb{N}$.

An operator $\Phi : \mathscr{F}_m \to \mathscr{F}_n$ is a total function.

Operator

Let us denote by $\mathscr{F}_n$ the class of all partial functions from $\mathbb{N}^n$ to $\mathbb{N}$.

An operator $\Phi : \mathscr{F}_m \to \mathscr{F}_n$ is a total function.

Question. In what sense is Φ computable?

Operator

1. $\Phi_1(f)(x) = 2f(x)$.

Operator

1. $\Phi_1(f)(x) = 2f(x)$.
2. $\Phi_2(f)(x) = \sum_{y \leq x} f(y)$.

Operator

1. $\Phi_1(f)(x) = 2f(x)$.
2. $\Phi_2(f)(x) = \sum_{y \leq x} f(y)$.

Calculations of $\Phi_1(f)$ and $\Phi_2(f)$, at say 3, only use a 'finite part' of the input function f.

Operator

1. $\Phi_1(f)(x) = 2f(x)$.

2. $\Phi_2(f)(x) = \sum_{y \leq x} f(y)$.

Calculations of $\Phi_1(f)$ and $\Phi_2(f)$, at say 3, only use a 'finite part' of the input function f.

If f is computable then both $\Phi_1(f)$ and $\Phi_2(f)$ are effective.

Finite Operator

A function θ is finite if its domain of definition is finite.

In this chapter θ always denotes a finite function.

Finite Operator

A function θ is finite if its domain of definition is finite.

In this chapter θ always denotes a finite function.

θ is a finite part of f if $\theta \subseteq f$.

Finite Operator

A finite function $\theta \in \mathscr{F}_n$ can be coded up by a number $\widetilde{\theta}$ in the following manner:

A tuple $\mathbf{x} = \langle x_1, \ldots, x_n \rangle$ is coded up by

$$\widetilde{\mathbf{x}} = p_1^{x_1+1} p_2^{x_2+1} \ldots p_n^{x_n+1}.$$

A finite function θ is coded up by

$$\begin{array}{rcll} \widetilde{\theta} & = & 0, & \text{if } dom(\theta) = \varnothing \\ \widetilde{\theta} & = & \prod_{\mathbf{x} \in dom(\theta)} p_{\widetilde{\mathbf{x}}}^{\theta(\mathbf{x})+1}, & \text{if } dom(\theta) \neq \varnothing. \end{array}$$

Monotonicity and Continuity

Suppose $\theta_0 \subseteq \theta_1 \subseteq \ldots \subseteq \theta_k \subseteq \ldots \subseteq f$ and $\bigcup_{i\in\omega} \theta_i = f$.

Monotonicity and Continuity

Suppose $\theta_0 \subseteq \theta_1 \subseteq \ldots \subseteq \theta_k \subseteq \ldots \subseteq f$ and $\bigcup_{i\in\omega} \theta_i = f$.

The monotonicity implies

$$\Phi(\theta_0) \subseteq \Phi(\theta_1) \subseteq \ldots \subseteq \Phi(\theta_k) \subseteq \ldots \subseteq \Phi(f).$$

The continuity says

$$\bigcup_{i\in\omega} \Phi(\theta_i) = \Phi(f).$$

Monotonicity and Continuity

Suppose $\theta_0 \subseteq \theta_1 \subseteq \ldots \subseteq \theta_k \subseteq \ldots \subseteq f$ and $\bigcup_{i\in\omega} \theta_i = f$.

The monotonicity implies

$$\Phi(\theta_0) \subseteq \Phi(\theta_1) \subseteq \ldots \subseteq \Phi(\theta_k) \subseteq \ldots \subseteq \Phi(f).$$

The continuity says

$$\bigcup_{i\in\omega} \Phi(\theta_i) = \Phi(f).$$

Two useful observations about $\bigcup_{i\in\omega} \Phi(\theta_i) = \Phi(f)$ are as follows:

1. All θ_i's are finite, thus computable, even if f is not computable.
2. To calculate $\Phi(f)(\mathbf{x})$, we only need to know $\Phi(\theta_i)(\mathbf{x})$ for some i.

Continuity

Let $\Phi : \mathscr{F}_m \rightarrow \mathscr{F}_n$ be an operator.

Φ is continuous if for any $f \in \mathscr{F}_m$ and all $\mathbf{x}, y$,

$$\Phi(f)(\mathbf{x}) \simeq y \text{ iff } \exists \theta \subseteq f.\Phi(\theta)(\mathbf{x}) \simeq y.$$

Φ is monotone if $\Phi(f) \subseteq \Phi(g)$ whenever $f \subseteq g \in \mathscr{F}_m$.

Continuity

Lemma. If Φ is continuous, then it is monotone.

Continuity

Lemma. If Φ is continuous, then it is monotone.

Proof. Suppose $f \subseteq g \in \mathscr{F}_m$. Then for all $\mathbf{x}, y$,

$$\begin{aligned} \Phi(f)(\mathbf{x}) \simeq y &\Leftrightarrow \exists \theta \subseteq f . \Phi(\theta)(\mathbf{x}) \simeq y \\ &\Rightarrow \exists \theta \subseteq g . \Phi(\theta)(\mathbf{x}) \simeq y \\ &\Leftrightarrow \Phi(g)(\mathbf{x}) \simeq y . \end{aligned}$$

□

Recursive Operator

$\Phi : \mathscr{F}_m \to \mathscr{F}_n$ is a recursive operator if there is a computable function $\phi(z, \mathbf{x})$ such that for all $f \in \mathscr{F}_m$ and $\mathbf{x} \in \mathbb{N}^n$, $y \in \mathbb{N}$,

$$\Phi(f)(\mathbf{x}) \simeq y \text{ iff } \exists \theta \subseteq f.\phi(\widetilde{\theta}, \mathbf{x}) \simeq y.$$

Recursive Operator

$\Phi(f) = 2f$ is a recursive operator. To see this define $\phi(z, x)$ by

$$\phi(z, x) = \begin{cases} 2\theta(x), & \text{if } z = \widetilde{\theta} \text{ and } x \in dom(\theta), \\ \uparrow, & \text{otherwise.} \end{cases}$$

Recursive Operator

Theorem. All recursive operators are continuous.

Recursive Operator

Theorem. All recursive operators are continuous.

Proof. If Φ is recursive, then to prove Φ is continuous is to prove that $\forall f \in \mathscr{F}_m, \mathbf{x} \in \mathbb{N}^n, y \in \mathbb{N}$, $\Phi(f)(\mathbf{x}) \simeq y$ iff $\exists \theta \subseteq f.\Phi(\theta)(\mathbf{x}) \simeq y$.

"$\Rightarrow$" Assume $\Phi(f)(\mathbf{x}) \simeq y$. Since Φ is recursive, there is a computable function $\phi(z, \mathbf{x})$ such that $\exists \theta \subseteq f.\phi(\widetilde{\theta}, \mathbf{x}) \simeq y$. Since $\theta \subseteq \theta$, by the definition of recursive operator, $\Phi(\theta)(\mathbf{x}) \simeq y$.

"$\Leftarrow$" Assume $\exists \theta \subseteq f.\Phi(\theta)(\mathbf{x}) \simeq y$. Since Φ is recursive, there is a computable function $\phi(z, \mathbf{x})$ such that $\exists \theta_1 \subseteq \theta.\phi(\widetilde{\theta_1}, \mathbf{x}) \simeq y$. Since $\theta_1 \subseteq \theta \subseteq f$, we have $\exists \theta_1 \subseteq f.\phi(\widetilde{\theta_1}, \mathbf{x}) \simeq y$. By the definition of recursive operator, $\Phi(f)(\mathbf{x}) \simeq y$. $\square$

Recursive Operator

Theorem. Let $\Phi : \mathscr{F}_m \rightarrow \mathscr{F}_n$ be an operator. Then Φ is a recursive operator iff

(1) Φ is continuous;

(2) the function $\varphi(z, \mathbf{x})$ given by

$$\varphi(z, \mathbf{x}) = \begin{cases} \Phi(\theta)(\mathbf{x}), & \text{if } z = \widetilde{\theta} \text{ for some } \theta \in \mathscr{F}_m, \\ \uparrow, & \text{otherwise}. \end{cases}$$

is computable.

Recursive Operator

Proof. Suppose Φ is recursive with computable function ϕ st.

$$\Phi(f)(\mathbf{x}) \simeq y \text{ iff } \exists\theta \subseteq f.\phi(\widetilde{\theta}, \mathbf{x}) \simeq y.$$

Let φ be given in the theorem. We have

$$\varphi(\widetilde{\theta}, \mathbf{x}) \simeq y \text{ iff } \Phi(\theta)(\mathbf{x}) \simeq y \text{ iff } \exists\theta_1 \subseteq \theta.\phi(\widetilde{\theta_1}, \mathbf{x}) \simeq y.$$

The rightmost is a partially decidable predicate.

Recursive Operator

Proof. Suppose Φ is recursive with computable function ϕ st.

$$\Phi(f)(\mathbf{x}) \simeq y \text{ iff } \exists\theta \subseteq f.\phi(\widetilde{\theta}, \mathbf{x}) \simeq y.$$

Let φ be given in the theorem. We have

$$\varphi(\widetilde{\theta}, \mathbf{x}) \simeq y \text{ iff } \Phi(\theta)(\mathbf{x}) \simeq y \text{ iff } \exists\theta_1 \subseteq \theta.\phi(\widetilde{\theta_1}, \mathbf{x}) \simeq y.$$

The rightmost is a partially decidable predicate.

Conversely suppose the conditions of the theorem hold. Then

$$\begin{aligned}\Phi(f)(\mathbf{x}) \simeq y \quad & \text{iff} \quad \exists\theta \subseteq f.\Phi(\theta)(\mathbf{x}) \simeq y \\ & \text{iff} \quad \exists\theta \subseteq f.\varphi(\widetilde{\theta}, \mathbf{x}) \simeq y.\end{aligned}$$

Recursive Operator

Corollary. Suppose $\Phi : \mathscr{F}_m \to \mathscr{F}_n$ is a recursive operator with the computable function ϕ. Then

$$\Phi(\theta)(\mathbf{x}) \simeq y \text{ iff } \phi(\widetilde{\theta}, \mathbf{x}) \simeq y.$$

Recursive Operator

Corollary. Suppose $\Phi : \mathscr{F}_m \to \mathscr{F}_n$ is a recursive operator with the computable function ϕ. Then

$$\Phi(\theta)(\mathbf{x}) \simeq y \text{ iff } \phi(\widetilde{\theta}, \mathbf{x}) \simeq y.$$

To show that an operator is recursive, it suffices to show that it

(1) is continuous; and

(2) is computable when restricted to the finite functions.

Examples of Recursive Operator

1. The diagonalisation operator $\Phi(f)(x) \simeq f(x, x)$.

Examples of Recursive Operator

1. The diagonalisation operator $\Phi(f)(x) \simeq f(x,x)$.
2. $\Phi(f)(x) \simeq \sum_{y \leq x} f(y)$.

Examples of Recursive Operator

1. The diagonalisation operator $\Phi(f)(x) \simeq f(x,x)$.
2. $\Phi(f)(x) \simeq \sum_{y \leq x} f(y)$.
3. $\Phi(f) \simeq g \circ f$, where g is computable.

Examples of Recursive Operator

1. The diagonalisation operator $\Phi(f)(x) \simeq f(x,x)$.
2. $\Phi(f)(x) \simeq \sum_{y \leq x} f(y)$.
3. $\Phi(f) \simeq g \circ f$, where g is computable.
4. The μ-operator $\Phi : \mathscr{F}_{n+1} \to \mathscr{F}_n$ given by

$$\Phi(f)(\mathbf{x}) \simeq \mu y(f(\mathbf{x}, y) = 0).$$

Second Order Computable Function

We know that the operations on computable functions is effective if they can be given by total computable functions acting on indices. E.g.,

$$\forall e \in \mathbb{N}, \exists \text{ total } g \in \mathscr{C}, (\phi_e)^2 = \phi_{g(e)}.$$

Second Order Computable Function

We know that the operations on computable functions is effective if they can be given by total computable functions acting on indices. E.g.,

$$\forall e \in \mathbb{N}, \exists \text{ total } g \in \mathscr{C}, (\phi_e)^2 = \phi_{g(e)}.$$

Now consider recursive operators. $\mathscr{F}_m$ is significantly larger than $\mathscr{C}_m$.

A recursive generator $\Phi : \mathscr{F}_m \to \mathscr{F}_n$ is essentially the second order recursive function $\Phi : \mathscr{C}_m \to \mathscr{C}_n$.

Second Order Computable Function

We know that the operations on computable functions is effective if they can be given by total computable functions acting on indices. E.g.,

$$\forall e \in \mathbb{N}, \exists \text{ total } g \in \mathscr{C}, (\phi_e)^2 = \phi_{g(e)}.$$

Now consider recursive operators. $\mathscr{F}_m$ is significantly larger than $\mathscr{C}_m$.

A recursive generator $\Phi : \mathscr{F}_m \to \mathscr{F}_n$ is essentially the second order recursive function $\Phi : \mathscr{C}_m \to \mathscr{C}_n$.

A total computable function h is extensional if, for all a, b, $\phi_{h(a)} = \phi_{h(b)}$ whenever $\phi_a = \phi_b$.

Myhill-Shepherdson Theorem

Theorem (Myhill-Shepherdson, Part I).

Suppose that $\Psi : \mathscr{F}_m \to \mathscr{F}_n$ is a recursive operator. Then there is a total extensional computable function h such that

$$\Psi(\phi_e^{(m)}) = \phi_{h(e)}^{(n)}.$$

Myhill-Shepherdson Theorem

Theorem (Myhill-Shepherdson, Part I).

Suppose that $\Psi : \mathscr{F}_m \to \mathscr{F}_n$ is a recursive operator. Then there is a total extensional computable function h such that

$$\Psi(\phi_e^{(m)}) = \phi_{h(e)}^{(n)}.$$

In other words, h is an index of the second order computable function Ψ.

Myhill-Shepherdson Theorem

Proof. By assumption, there is a computable function ψ such that

$$\Psi(\phi_e^{(m)})(\mathbf{x}) \simeq y \text{ iff } \exists\theta \subseteq \phi_e^{(m)}.\psi(\widetilde{\theta}, \mathbf{x}) \simeq y.$$

Myhill-Shepherdson Theorem

Proof. By assumption, there is a computable function ψ such that

$$\Psi(\phi_e^{(m)})(\mathbf{x}) \simeq y \text{ iff } \exists\theta \subseteq \phi_e^{(m)}.\psi(\widetilde{\theta}, \mathbf{x}) \simeq y.$$

Let $R(z, e, \mathbf{x}, y)$ be given by

$$R(z, e, \mathbf{x}, y) \simeq \exists\theta.(z = \widetilde{\theta} \wedge \theta \subseteq \phi_e^{(m)} \wedge \psi(\widetilde{\theta}, \mathbf{x}) \simeq y).$$

Myhill-Shepherdson Theorem

Proof. By assumption, there is a computable function ψ such that

$$\Psi(\phi_e^{(m)})(\mathbf{x}) \simeq y \text{ iff } \exists\theta \subseteq \phi_e^{(m)}.\psi(\widetilde{\theta}, \mathbf{x}) \simeq y.$$

Let $R(z, e, \mathbf{x}, y)$ be given by

$$R(z, e, \mathbf{x}, y) \simeq \exists\theta.(z = \widetilde{\theta} \wedge \theta \subseteq \phi_e^{(m)} \wedge \psi(\widetilde{\theta}, \mathbf{x}) \simeq y).$$

It is easy to see that $R(z, e, \mathbf{x}, y)$ is partially decidable.

Myhill-Shepherdson Theorem

Proof. By assumption, there is a computable function ψ such that

$$\Psi(\phi_e^{(m)})(\mathbf{x}) \simeq y \text{ iff } \exists \theta \subseteq \phi_e^{(m)}.\psi(\widetilde{\theta}, \mathbf{x}) \simeq y.$$

Let $R(z, e, \mathbf{x}, y)$ be given by

$$R(z, e, \mathbf{x}, y) \simeq \exists \theta.(z = \widetilde{\theta} \wedge \theta \subseteq \phi_e^{(m)} \wedge \psi(\widetilde{\theta}, \mathbf{x}) \simeq y).$$

It is easy to see that $R(z, e, \mathbf{x}, y)$ is partially decidable.

So $\exists z.R(z, e, \mathbf{x}, y)$ is partially decidable.

Myhill-Shepherdson Theorem

It follows from

$$\Psi(\phi_e^{(m)})(\mathbf{x}) \simeq y \text{ iff } \exists z.R(z, e, \mathbf{x}, y)$$

that $\Psi(\phi_e^{(m)})(\mathbf{x}) \simeq y$ is partially decidable. Thus $\Psi(\phi_e^{(m)})(\mathbf{x})$ is computable.

Myhill-Shepherdson Theorem

It follows from

$$\Psi(\phi_e^{(m)})(\mathbf{x}) \simeq y \text{ iff } \exists z.R(z, e, \mathbf{x}, y)$$

that $\Psi(\phi_e^{(m)})(\mathbf{x}) \simeq y$ is partially decidable. Thus $\Psi(\phi_e^{(m)})(\mathbf{x})$ is computable.

By s-m-n Theorem, there is a total computable function h s.t.

$$\phi_{h(e)}^{(n)}(\mathbf{x}) \simeq \Psi(\phi_e^{(m)})(\mathbf{x})$$

for all e.

Myhill-Shepherdson Theorem

It follows from

$$\Psi(\phi_e^{(m)})(\mathbf{x}) \simeq y \text{ iff } \exists z.R(z, e, \mathbf{x}, y)$$

that $\Psi(\phi_e^{(m)})(\mathbf{x}) \simeq y$ is partially decidable. Thus $\Psi(\phi_e^{(m)})(\mathbf{x})$ is computable.

By s-m-n Theorem, there is a total computable function h s.t.

$$\phi_{h(e)}^{(n)}(\mathbf{x}) \simeq \Psi(\phi_e^{(m)})(\mathbf{x})$$

for all e.

Clearly the function h must be extensional.

Myhill-Shepherdson Theorem

Theorem (Myhill-Shepherdson, Part II).

Suppose that h is a total extensional computable function. Then there is a unique recursive operator Ψ such that

$$\Psi(\phi_e^{(m)}) = \phi_{h(e)}^{(n)}$$

for all e.

Myhill-Shepherdson Theorem

Proof. The extensionality condition allows one to define

$$\Psi_0(\phi_e) = \phi_{h(e)}.$$

The operator Ψ that extends Ψ_0 can be defined as follows:

$$\Psi(f)(\mathbf{x}) \simeq y \text{ if } \exists\theta \subseteq f.\Psi_0(\theta)(\mathbf{x}) \simeq y. \tag{1}$$

Myhill-Shepherdson Theorem

Proof. The extensionality condition allows one to define

$$\Psi_0(\phi_e) = \phi_{h(e)}.$$

The operator Ψ that extends Ψ_0 can be defined as follows:

$$\Psi(f)(\mathbf{x}) \simeq y \text{ if } \exists\theta \subseteq f.\Psi_0(\theta)(\mathbf{x}) \simeq y. \tag{1}$$

We claim that for every computable function f,

$$\Psi_0(f)(\mathbf{x}) \simeq y \text{ iff } \exists\theta \subseteq f.\Psi_0(\theta)(\mathbf{x}) \simeq y. \tag{2}$$

Myhill-Shepherdson Theorem

Proof. The extensionality condition allows one to define

$$\Psi_0(\phi_e) = \phi_{h(e)}.$$

The operator Ψ that extends Ψ_0 can be defined as follows:

$$\Psi(f)(\mathbf{x}) \simeq y \text{ if } \exists \theta \subseteq f . \Psi_0(\theta)(\mathbf{x}) \simeq y. \quad (1)$$

We claim that for every computable function f,

$$\Psi_0(f)(\mathbf{x}) \simeq y \text{ iff } \exists \theta \subseteq f . \Psi_0(\theta)(\mathbf{x}) \simeq y. \quad (2)$$

It is clear that (2) implies the well-definedness of (1), which in turn implies that Ψ is continuous and is unqiue.

Myhill-Shepherdson Theorem

Now we prove the claim that

$$\Psi_0(f)(\mathbf{x}) \simeq y \text{ iff } \exists \theta \subseteq f.\Psi_0(\theta)(\mathbf{x}) \simeq y.$$

Myhill-Shepherdson Theorem

Now we prove the claim that

$$\Psi_0(f)(\mathbf{x}) \simeq y \text{ iff } \exists\theta \subseteq f.\Psi_0(\theta)(\mathbf{x}) \simeq y.$$

Let $\mathscr{A}$ be $\{f \in \mathscr{C}_m \mid \Psi_0(f)(\mathbf{x}) \simeq y\}$. Then $A = \{e \mid \phi_e \in \mathscr{A}\}$ is r.e.

Myhill-Shepherdson Theorem

Now we prove the claim that

$$\Psi_0(f)(\mathbf{x}) \simeq y \text{ iff } \exists \theta \subseteq f . \Psi_0(\theta)(\mathbf{x}) \simeq y.$$

Let $\mathscr{A}$ be $\{f \in \mathscr{C}_m \mid \Psi_0(f)(\mathbf{x}) \simeq y\}$. Then $A = \{e \mid \phi_e \in \mathscr{A}\}$ is r.e.

So by Rice-Shapiro Theorem, if f is computable then

$$f \in \mathscr{A} \text{ iff } \exists \theta \subseteq f . \theta \in \mathscr{A},$$

which is precisely the above equivalence.

Myhill-Shepherdson Theorem

Next we show that the function

$$\psi(z, \mathbf{x}) = \begin{cases} \Psi(\theta)(\mathbf{x}), & \text{if } z = \widetilde{\theta} \text{ for some } \theta, \\ \uparrow, & \text{otherwise.} \end{cases}$$

is computable.

Myhill-Shepherdson Theorem

Next we show that the function

$$\psi(z, \mathbf{x}) = \begin{cases} \Psi(\theta)(\mathbf{x}), & \text{if } z = \widetilde{\theta} \text{ for some } \theta, \\ \uparrow, & \text{otherwise.} \end{cases}$$

is computable.

By Church's Thesis, there is a total computable function c such that $c(\widetilde{\theta})$ is an index for θ.

Myhill-Shepherdson Theorem

Next we show that the function

$$\psi(z, \mathbf{x}) = \begin{cases} \Psi(\theta)(\mathbf{x}), & \text{if } z = \widetilde{\theta} \text{ for some } \theta, \\ \uparrow, & \text{otherwise.} \end{cases}$$

is computable.

By Church's Thesis, there is a total computable function c such that $c(\widetilde{\theta})$ is an index for θ.

Therefore $\psi(\widetilde{\theta}, \mathbf{x}) \simeq \Psi(\phi_{c(\widetilde{\theta})})(\mathbf{x}) \simeq \phi_{h(c(\widetilde{\theta}))}(\mathbf{x})$. $\square$

First Recursion Theorem, Kleene

The First Recursion Theorem (The Fixpoint Theorem).

Suppose that $\Phi : \mathscr{F}_m \to \mathscr{F}_m$ is a recursive operator. Then there is a computable function f_Φ that is the least fixpoint of Φ, i.e.

- $\Phi(f_\Phi) = f_\Phi$;
- if $\Phi(g) = g$, then $f_\Phi \subseteq g$.

First Recursion Theorem, Kleene

The First Recursion Theorem (The Fixpoint Theorem).

Suppose that $\Phi : \mathscr{F}_m \to \mathscr{F}_m$ is a recursive operator. Then there is a computable function f_Φ that is the least fixpoint of Φ, i.e.

- $\Phi(f_\Phi) = f_\Phi$;
- if $\Phi(g) = g$, then $f_\Phi \subseteq g$.

Proof. Continuity is responsible for the existence of the least fixpoint f_Φ constructed as the limit of the increasing sequence

$$f_\varnothing \subseteq \Phi(f_\varnothing) \subseteq \Phi(\Phi(f_\varnothing)) \subseteq \ldots \subseteq \Phi^i(f_\varnothing) \subseteq \ldots.$$

Let f_i be $\Phi^i(f_\varnothing)$. Note $f_0 = f_\varnothing$.

First Recursion Theorem, Kleene

Since Φ is recursive, $\Phi(\phi_e) = \phi_{h(e)}$ for some total extensional computable function h.

First Recursion Theorem, Kleene

Since Φ is recursive, $\Phi(\phi_e) = \phi_{h(e)}$ for some total extensional computable function h.

Let e_0 be an index for f_0. Define a computable function k by

$$\begin{array}{rcl} k(0) & = & e_0, \\ k(n+1) & = & h(k(n)). \end{array}$$

Then $f_n = \phi_{k(n)}$. Now

$$f_\Phi(\mathbf{x}) \simeq y \text{ iff } \exists n.\phi_{k(n)}(\mathbf{x}) \simeq y.$$

The relation on the right hand is partially decidable. We conclude that f_Φ is computable. □

An Example

Let Φ be the recursive operator given by

$$\begin{aligned}\Phi(f)(0) &= 1,\\ \Phi(f)(n+1) &= f(n+2).\end{aligned}$$

The least fixpoint is

$$\begin{aligned}\Phi(f)(0) &= 1,\\ \Phi(f)(n+1) &= \uparrow.\end{aligned}$$

Other fixpoints take the form

$$\begin{aligned}\Phi(f)(0) &= 1,\\ \Phi(f)(n+1) &= a.\end{aligned}$$

Ackermann Operator

The Ackermann operator: Let $\Phi : \mathscr{F}_2 \to \mathscr{F}_2$ be given by

$$\begin{array}{rcl} \Phi(f)(0,y) & = & y+1, \\ \Phi(f)(x+1,0) & = & f(x+1), \\ \Phi(f)(x+1,y+1) & = & f(x,f(x+1,y)). \end{array}$$

Φ is a recursive operator.

Ackermann Operator

The Ackermann operator: Let $\Phi : \mathscr{F}_2 \to \mathscr{F}_2$ be given by

$$\begin{aligned}
\Phi(f)(0,y) &= y+1, \\
\Phi(f)(x+1,0) &= f(x+1), \\
\Phi(f)(x+1,y+1) &= f(x,f(x+1,y)).
\end{aligned}$$

Φ is a recursive operator.

The unique fixpoint of this operator is the Ackermann function.

Some Note

Recursive Definition: The First Recursion Theorem says that for a recursive operator Φ of type $\mathscr{F}_m \to \mathscr{F}_m$, we may think of

$$f = \Phi(f) \tag{3}$$

as a recursive definition.

The least fixpoint of the operator is the computable function defined by the general recursion (3).

Some Note

Recursive Definition: The First Recursion Theorem says that for a recursive operator Φ of type $\mathscr{F}_m \to \mathscr{F}_m$, we may think of

$$f = \Phi(f) \tag{3}$$

as a recursive definition.

The least fixpoint of the operator is the computable function defined by the general recursion (3).

For Computable Function: Suppose h is a total extensional computable function. Then there is some n such that

$$\phi_n \simeq \phi_{h(n)}.$$

Moreover $\phi_n \subseteq \phi_m$ whenever $\phi_m \simeq \phi_{h(m)}$.

Outline

1. First Recursion Theorem
   - Recursive Operator
   - Second Order Computable Function
   - First Recursion Theorem
2. Second Recursion Theorem
   - Second Recursion Theorem
   - The Diagonal Argument

The Second Recursion Theorem

The Second Recursion Theorem.

Let f be a total unary computable function. Then there is a number n such that $\phi_{f(n)} = \phi_n$.

The Second Recursion Theorem

The Second Recursion Theorem.

Let f be a total unary computable function. Then there is a number n such that $\phi_{f(n)} = \phi_n$.

Proof. By the s-m-n theorem there is a *total* computable function $s(x)$ such that for all x

$$\phi_{f(\phi_x(x))}(y) \simeq \phi_{s(x)}(y). \tag{4}$$

Let m be such that $s = \phi_m$. Rewriting (4) we have

$$\phi_{f(\phi_x(x))}(y) \simeq \phi_{\phi_m(x)}(y).$$

We are done by letting x be m and n be $\phi_m(m)$. □

Comment on the Second Recursion Theorem

The Second Recursion Theorem is not really about fixpoint.
The operator

$$\Phi : \phi_x \mapsto \phi_{f(x)}$$

is not well-defined if f is not extensional.

Comment on the Second Recursion Theorem

The Second Recursion Theorem is not really about fixpoint.
The operator

$$\Phi : \phi_x \mapsto \phi_{f(x)}$$

is not well-defined if f is not extensional.

But we do have the following induced mapping of programs

$$f^*(P_x) = P_{f(x)}.$$

Some Corollaries

Corollary. If f is a total computable function, there is a number n such that $W_{f(n)} = W_n$ and $E_{f(n)} = E_n$.

Some Corollaries

Corollary. If f is a total computable function, there are arbitrarily large numbers n such that $\phi_{f(n)} = \phi_n$.

Some Corollaries

Corollary. If f is a total computable function, there are arbitrarily large numbers n such that $\phi_{f(n)} = \phi_n$.

Proof. Given any number k, pick up a number e such that

$$\phi_e \neq \phi_0, \phi_1, \ldots, \phi_k.$$

Now define a function g by

$$g(x) = \begin{cases} e, & \text{if } x \leq k, \\ f(x), & \text{if } x > k. \end{cases}$$

A fixpoint of g must be greater than k, and consequently it is also a fixpoint of f. □

Some Corollaries

Corollary. Let $f(x,y)$ be a computable function. Then there is an index e such that

$$\phi_e(y) \simeq f(e,y).$$

Some Corollaries

Corollary. Let $f(x, y)$ be a computable function. Then there is an index e such that

$$\phi_e(y) \simeq f(e, y).$$

Proof. By s-m-n Theorem there is a total computable function $s(x)$ such that $\phi_{s(x)}(y) \simeq f(x, y)$. We are done by applying the Second Recursion Theorem. □

Some Corollaries

Corollary. Let $f(x,y)$ be a computable function. Then there is an index e such that

$$\phi_e(y) \simeq f(e,y).$$

Proof. By s-m-n Theorem there is a total computable function $s(x)$ such that $\phi_{s(x)}(y) \simeq f(x,y)$. We are done by applying the Second Recursion Theorem. $\square$

Remark. The above corollary makes it meaningful to define a computable function $\phi_e(y)$ by a computable function $f(e,y)$.

Some Corollaries

There is a number n such that $\phi_n(x) = x^n$.

Some Corollaries

There is a number n such that $\phi_n(x) = x^n$.

There is a number n such that $W_n = \{n\}$. This number is obtained by applying the above corollary to the function

$$f(x,y) = \begin{cases} 0, & \text{if } x = y, \\ \text{undefined}, & \text{otherwise}. \end{cases}$$

Some Corollaries

Corollary. There is a program P such that for all x, $P(x) \downarrow \gamma(P)$.
($\gamma(P)$ is the Gödel number of program P.)

Some Corollaries

Corollary. There is a program P such that for all x, $P(x) \downarrow \gamma(P)$.
($\gamma(P)$ is the Gödel number of program P.)

Proof. The theorem says that there is a number n such that

$$\phi_n(x) = n$$

for all x. Simply apply one of the corollaries to $f(m, x) = m$. $\square$

Some Negative Results

Theorem. K is not recursive.

Some Negative Results

Theorem. K is not recursive.

Proof. Let a, b be indices such that $W_a = \varnothing$ and $W_b = \mathbb{N}$. If K were recursive then the function

$$g(x) = \begin{cases} a, & \text{if } x \in K, \\ b, & \text{if } x \notin K, \end{cases}$$

would be computable. Notice that for every x

$$W_{g(x)} \neq W_x,$$

which would contradict to the Second Recursion Theorem. $\square$

Some Negative Results

Rice Theorem. Suppose $\emptyset \subsetneq \mathscr{A} \subsetneq \mathscr{C}_1$ and $A = \{x \mid \phi_x \in \mathscr{A}\}$. Then A is not recursive.

Some Negative Results

Rice Theorem. Suppose $\varnothing \subsetneq \mathscr{A} \subsetneq \mathscr{C}_1$ and $A = \{x \mid \phi_x \in \mathscr{A}\}$. Then A is not recursive.

Proof. Let $a \in A$ and $b \notin A$. If A were recursive, then the function f given by

$$f(x) = \begin{cases} a, & \text{if } x \notin A, \\ b, & \text{if } x \in A, \end{cases}$$

would be computable. By definition $x \in A$ iff $f(x) \notin A$. By the Second Recursion Theorem there would be some n such that $\phi_{f(n)} = \phi_n$. So $n \in A$ iff $f(n) \in A$, which led to a contradiction. □

Some Negative Results

Theorem. Suppose that f is a total increasing function such that

- if $m \neq n$ then $\phi_{f(m)} \neq \phi_{f(n)}$,
- $f(n)$ is the least index of the function $\phi_{f(n)}$.

Then f is not computable.

Some Negative Results

Theorem. Suppose that f is a total increasing function such that

- if $m \neq n$ then $\phi_{f(m)} \neq \phi_{f(n)}$,
- $f(n)$ is the least index of the function $\phi_{f(n)}$.

Then f is not computable.

Proof. Suppose f satisfies the conditions of the theorem.
By the first condition $f(n) > n$ if n is large enough.
By the second condition $\phi_{f(n)} \neq \phi_n$ for all large enough n.
This contradicts to one of the corollaries. □

The First vs. The Second

The First Recursion Theorem defines a program.

The Second Recursion Theorem offers a computable function.

The Diagonal Argument

If h is total, then $\phi_{h(x)}$ can be regarded as an effective enumerator of the computable functions

$$\phi_{h(0)}, \phi_{h(1)}, \phi_{h(2)}, \ldots, \phi_{h(i)}, \ldots.$$

If h is not total, we identify $\phi_{h(x)}(y)$ to $\psi_U(h(x), y)$.

The Diagonal Argument

Lemma. Suppose that h is a computable function. There is a total computable function h' such that h and h' enumerate the same sequence of computable functions.

The Diagonal Argument

Lemma. Suppose that h is a computable function. There is a total computable function h' such that h and h' enumerate the same sequence of computable functions.

Proof. By s-m-n Theorem, there is a total function h' such that $\phi_{h'(x)}(y) \simeq \psi_U(h(x), y)$.

The Diagonal Argument

Let's denote by $\mathbf{E}_k$ the sequence of computable functions effectively enumerated by ϕ_k.

The Diagonal Argument

Let's denote by $\mathbf{E}_k$ the sequence of computable functions effectively enumerated by ϕ_k.

The diagonal enumeration $\mathbf{D}$ is

$$\phi_{\phi_0(0)}, \phi_{\phi_1(1)}, \phi_{\phi_2(2)}, \cdots,$$

given by the function $h(x) \simeq \phi_x(x)$.

The Diagonal Argument

Let's denote by $\mathbf{E}_k$ the sequence of computable functions effectively enumerated by ϕ_k.

The diagonal enumeration $\mathbf{D}$ is

$$\phi_{\phi_0(0)}, \phi_{\phi_1(1)}, \phi_{\phi_2(2)}, \cdots,$$

given by the function $h(x) \simeq \phi_x(x)$.

Suppose f is a total computable function. The enumeration $\mathbf{D}^*$ is

$$\phi_{f(\phi_0(0))}, \phi_{f(\phi_1(1))}, \phi_{f(\phi_2(2))}, \cdots.$$

The Diagonal Argument

Let's denote by $\mathbf{E}_k$ the sequence of computable functions effectively enumerated by ϕ_k.

The diagonal enumeration $\mathbf{D}$ is

$$\phi_{\phi_0(0)}, \phi_{\phi_1(1)}, \phi_{\phi_2(2)}, \cdots,$$

given by the function $h(x) \simeq \phi_x(x)$.

Suppose f is a total computable function. The enumeration $\mathbf{D}^*$ is

$$\phi_{f(\phi_0(0))}, \phi_{f(\phi_1(1))}, \phi_{f(\phi_2(2))}, \cdots.$$

By the lemma we may assume that $\mathbf{D}^* = \mathbf{E}_m$ for some total function ϕ_m.

The Diagonal Argument

Let's denote by $\mathbf{E}_k$ the sequence of computable functions effectively enumerated by ϕ_k.

The diagonal enumeration $\mathbf{D}$ is

$$\phi_{\phi_0(0)}, \phi_{\phi_1(1)}, \phi_{\phi_2(2)}, \cdots,$$

given by the function $h(x) \simeq \phi_x(x)$.

Suppose f is a total computable function. The enumeration $\mathbf{D}^*$ is

$$\phi_{f(\phi_0(0))}, \phi_{f(\phi_1(1))}, \phi_{f(\phi_2(2))}, \cdots.$$

By the lemma we may assume that $\mathbf{D}^* = \mathbf{E}_m$ for some total function ϕ_m.

So $\phi_{\phi_m(m)} = \phi_{f(\phi_m(m))}$. $\square$

The Diagonal Argument

The above is a diagonal argument.

The Diagonal Argument

The above is a diagonal argument.

The Second Recursion Theorem can be viewed as a generalization of many diagonal arguments.

Generalizing the Second Recursion Theorem

Second Recursion Theorem. Suppose $f(x,z)$ is a total computable function. There is a total computable function $n(z)$ such that for all z

$$\phi_{f(n(z),z)} = \phi_{n(z)}.$$

Generalizing the Second Recursion Theorem

Second Recursion Theorem. Suppose $f(x,z)$ is a total computable function. There is a total computable function $n(z)$ such that for all z

$$\phi_{f(n(z),z)} = \phi_{n(z)}.$$

Proof. By s-m-n Theorem there is a total computable function $s(x,z)$ such that

$$\phi_{f(\phi_x(x),z)} = \phi_{s(x,z)}.$$

By the same theorem there is a total computable function $m(z)$ such that $s(x,z) = \phi_{m(z)}(x)$. So

$$\phi_{f(\phi_x(x),z)} = \phi_{\phi_{m(z)}(x)}.$$

We are done by letting $x = m(z)$ and $n(z) = \phi_{m(z)}(m(z))$. □