Computable Functions

- In a formal theory of computability, every problem instance can be represented by a number and every number represents a problem instance.
- A problem is a function $f : \mathbb{N} \to \mathbb{N}$ from numbers to numbers.
- A problem is computable if it can be calculated by a program.

Decision Problem

A problem $f : \mathbb{N} \to \mathbb{N}$ is a decision problem if the range $ran(f)$ of f is $\{0, 1\}$, where 1 denotes a 'yes' answer and 0 a 'no' answer.

A decision problem g can be identified with the set $\{n \mid g(n) = 1\}$.

Conversely a subset A of $\mathbb{N}$ can be seen as a decision problem via the characteristic function of A:

$$c_A(n) = \begin{cases} 1, & \text{if } x \in A, \\ 0, & \text{otherwise.} \end{cases}$$

Decision Problem as Predicate

A decision problem can be stated as a predicate $P(x)$ on number.

It relates to the problem-as-function viewpoint by the following characteristic function of $P(x)$:

$$c_P(n) = \begin{cases} 1, & \text{if } P(n) \text{ is valid,} \\ 0, & \text{otherwise.} \end{cases}$$

$$\begin{aligned} \text{Decision Problem} &\Leftrightarrow \text{Subset of } \mathbb{N} \\ &\Leftrightarrow \text{Predicate on } \mathbb{N} \end{aligned}$$

Unlimited Register Machine Model

The Unlimited Register Machine Model belongs to the CM class.

Computability and Recursive Functions, by J. Shepherdson and H. Sturgis, in Journal of Symbolic Logic (**32**):1-63, 1965.

Instruction

Type	Instruction	Response of the URM
Zero	$Z(n)$	Replace r_n by 0.
Successor	$S(n)$	Add 1 to r_n.
Transfer	$T(m,n)$	Copy r_m to R_n.
Jump	$J(m,n,q)$	If $r_m = r_n$, go to the q-th instruction; otherwise go to the next instruction.

URM-Computable Function

Suppose P is the program of a URM and $a_1, \ldots, a_n, b \in \mathbb{N}$.

URM-Computable Function

Suppose P is the program of a URM and $a_1, \ldots, a_n, b \in \mathbb{N}$.

The computation $P(a_1, \ldots, a_n)$ converges to b if $P(a_1, \ldots, a_n) \downarrow$ and $r_1 = b$ in the final configuration.

URM-Computable Function

Suppose P is the program of a URM and $a_1, \ldots, a_n, b \in \mathbb{N}$.

The computation $P(a_1, \ldots, a_n)$ converges to b if $P(a_1, \ldots, a_n) \downarrow$ and $r_1 = b$ in the final configuration.

In this case we write $P(a_1, \ldots, a_n) \downarrow b$.

URM-Computable Function

Suppose P is the program of a URM and $a_1, \ldots, a_n, b \in \mathbb{N}$.

The computation $P(a_1, \ldots, a_n)$ converges to b if $P(a_1, \ldots, a_n) \downarrow$ and $r_1 = b$ in the final configuration.

In this case we write $P(a_1, \ldots, a_n) \downarrow b$.

P URM-computes f if, for all $a_1, \ldots, a_n, b \in \mathbb{N}$, $P(a_1, \ldots, a_n) \downarrow b$ iff $f(a_1, \ldots, a_n) = b$.

URM-Computable Function

Suppose P is the program of a URM and $a_1, \ldots, a_n, b \in \mathbb{N}$.

The computation $P(a_1, \ldots, a_n)$ converges to b if $P(a_1, \ldots, a_n) \downarrow$ and $r_1 = b$ in the final configuration.

In this case we write $P(a_1, \ldots, a_n) \downarrow b$.

P URM-computes f if, for all $a_1, \ldots, a_n, b \in \mathbb{N}$, $P(a_1, \ldots, a_n) \downarrow b$ iff $f(a_1, \ldots, a_n) = b$.

The function f is URM-definable if there is a program that URM-computes f.

We shall abbreviate “URM-computable” to “computable”.

Let

$$\mathscr{C}$$

be the set of computable functions and

$$\mathscr{C}_n$$

be the set of n-ary computable functions.

Some Notations

The notation $P[I_1, \ldots, I_n \to I]$ stands for the following program

$$
\begin{array}{rcl}
I_1 & : & T(I_1, 1) \\
 & \vdots & \\
I_n & : & T(I_n, n) \\
I_{n+1} & : & Z(n+1) \\
 & \vdots & \\
I_{\rho(P)} & : & Z(\rho(P)) \\
\_ & : & P \\
\_ & : & T(1, I)
\end{array}
$$

Primitive Recursive Recursion

The set of primitive recursive function is the least set generated from the initial functions, composition and recursion.

Bounded Sum and Bounded Product

Bounded sum:

$$\begin{aligned} \sum_{y<0} f(\widetilde{x}, y) &= 0, \\ \sum_{y<z+1} f(\widetilde{x}, y) &= \sum_{y<z} f(\widetilde{x}, y) + f(\widetilde{x}, z). \end{aligned}$$

Bounded product:

$$\begin{aligned} \prod_{y<0} f(\widetilde{x}, y) &= 1, \\ \prod_{y<z+1} f(\widetilde{x}, y) &= (\prod_{y<z} f(\widetilde{x}, y)) \cdot f(\widetilde{x}, z). \end{aligned}$$

Bounded Sum and Bounded Product

By composition the following functions are also primitive recursive if $k(\widetilde{x}, \widetilde{w})$ is primitive recursive:

$$\sum_{z<k(\widetilde{x},\widetilde{w})} f(\widetilde{x}, z)$$

and

$$\prod_{z<k(\widetilde{x},\widetilde{w})} f(\widetilde{x}, z).$$

Bounded Minimization Operator

Bounded search:

$$\mu z{<}y(f(\widetilde{x}, z) = 0) \stackrel{\text{def}}{=} \begin{cases} \text{the least } z < y, & \text{such that } f(\widetilde{x}, z) = 0; \\ y, & \text{if there is no such } z. \end{cases}$$

Bounded Minimization Operator

If $f(\widetilde{x}, z)$ and $k(\widetilde{x}, \widetilde{w})$ are primitive recursive functions, then so is the function

$$\mu z{<}k(\widetilde{x}, \widetilde{w})(f(\widetilde{x}, z) = 0).$$

Primitive Recursive Predicate

Suppose $M(x_1, \ldots, x_n)$ is an n-ary predicate of natural numbers. The characteristic function $c_M(\widetilde{x})$, where $\widetilde{x} = x_1, \ldots, x_n$, is

$$c_M(a_1, \ldots, a_n) = \begin{cases} 1, & \text{if } M(a_1, \ldots, a_n) \text{ holds,} \\ 0, & \text{if otherwise.} \end{cases}$$

The predicate $M(\widetilde{x})$ is primitive recursive if c_M is primitive recursive.

Closure Property

Proposition

The following statements are valid:

- If $R(\widetilde{x})$ is a primitive recursive predicate, then so is $\neg R(\widetilde{x})$.
- If $R(\widetilde{x})$, $S(\widetilde{x})$ are primitive recursive predicates, then the following predicates are primitive recursive:
  - $R(\widetilde{x}) \wedge S(\widetilde{x})$;
  - $R(\widetilde{x}) \vee S(\widetilde{x})$.
- If $R(\widetilde{x}, y)$ is a primitive recursive predicate, then the following predicates are primitive recursive:
  - $\forall z < y.R(\widetilde{x}, z)$;
  - $\exists z < y.R(\widetilde{x}, z)$.

Definition by Case

Proposition

Suppose that $f_1(\widetilde{x}), \ldots, f_k(\widetilde{x})$ are primitive recursive functions, and $M_1(\widetilde{x}), \ldots, M_k(\widetilde{x})$ are primitive recursive predicates, such that for every $\widetilde{x}$ exactly one of $M_1(\widetilde{x}), \ldots, M_k(\widetilde{x})$ holds. Then the function $g(\widetilde{x})$ given by

$$g(\widetilde{x}) = \begin{cases} f_1(\widetilde{x}), & \text{if } M_1(\widetilde{x}) \text{ holds,} \\ f_2(\widetilde{x}), & \text{if } M_2(\widetilde{x}) \text{ holds,} \\ \vdots & \\ f_k(\widetilde{x}), & \text{if } M_k(\widetilde{x}) \text{ holds.} \end{cases}$$

is primitive recursive.

More Arithmetic Functions

The following functions are primitive recursive.

1. $D(x) =$ the number of divisors of x;
2. $Pr(x) = \begin{cases} 1, & \text{if } x \text{ is prime,} \\ 0, & \text{if } x \text{ is not prime.} \end{cases}$
3. $p_x =$ the x-th prime number;
4. $(x)_y = \begin{cases} k, & k \text{ is the exponent of } p_y \text{ in the prime} \\ & \text{factorisation of } x, \text{for } x, y > 0, \\ 0, & \text{if } x = 0 \text{ or } y = 0. \end{cases}$

Not all Computable Functions are Primitive Recursive

Using the fact that all primitive recursive functions are total, a diagonalisation argument shows that non-primitive recursive computable functions must exist.

Minimization Operator, or Search Operator

Minimization function, or μ-function, or search function:

$$\mu y(f(\widetilde{x}, y) = 0) \quad \simeq \quad \begin{cases} \text{the least } y \text{ such that} \\ \quad f(\widetilde{x}, z) \text{ is defined for all } z \leq y, \text{ and} \\ \quad f(\widetilde{x}, y) = 0, \\ \text{undefined if otherwise.} \end{cases}$$

Here $\simeq$ is the computational equality.

Recursive Function

The set of recursive functions is the least set generated from the initial functions, composition, recursion and minimization.

Decidable Predicate

A predicate $R(\widetilde{x})$ is decidable if its characteristic function

$$c_R(\widetilde{x}) \stackrel{\text{def}}{=} \begin{cases} 1, & \text{if } R(\widetilde{x}) \text{ is true,} \\ 0, & \text{otherwise.} \end{cases}$$

is a recursive function.

Decidable Predicate

A predicate $R(\widetilde{x})$ is decidable if its characteristic function

$$c_R(\widetilde{x}) \stackrel{\text{def}}{=} \begin{cases} 1, & \text{if } R(\widetilde{x}) \text{ is true,} \\ 0, & \text{otherwise.} \end{cases}$$

is a recursive function. The predicate $R(\widetilde{x})$ is partially decidable if its partial characteristic function

$$\chi_R(\widetilde{x}) \stackrel{\text{def}}{=} \begin{cases} 1, & \text{if } R(\widetilde{x}) \text{ is true,} \\ \uparrow, & \text{otherwise.} \end{cases}$$

is a recursive function.

Closure Property

The following statements are valid:

- If $R(\widetilde{x})$ is decidable, then so is $\neg R(\widetilde{x})$.
- If $R(\widetilde{x}), S(\widetilde{x})$ are (partially) decidable, then the following predicates are (partially) decidable:
  - $R(\widetilde{x}) \wedge S(\widetilde{x})$;
  - $R(\widetilde{x}) \vee S(\widetilde{x})$.
- If $R(\widetilde{x}, y)$ is (partially) decidable, then the following predicates are (partially) decidable:
  - $\forall z < y.R(\widetilde{x}, y)$;
  - $\exists z < y.R(\widetilde{x}, y)$.

Definition by Cases

Suppose $f_1(\widetilde{x}), \ldots, f_k(\widetilde{x})$ are recursive and $M_1(\widetilde{x}), \ldots, M_k(\widetilde{x})$ are partially decidable. For every $\widetilde{x}$ at most one of $M_1(\widetilde{x}), \ldots, M_k(\widetilde{x})$ holds. Then the function $g(\widetilde{x})$ given by

$$g(\widetilde{x}) \simeq \begin{cases} f_1(\widetilde{x}), & \text{if } M_1(\widetilde{x}) \text{ holds,} \\ f_2(\widetilde{x}), & \text{if } M_2(\widetilde{x}) \text{ holds,} \\ \vdots & \\ f_k(\widetilde{x}), & \text{if } M_k(\widetilde{x}) \text{ holds.} \end{cases}$$

is recursive.

Minimization via Decidable Predicate

Suppose $R(x, y)$ is a partially decidable predicate. The function

$$
\begin{aligned}
g(x) &= \mu y R(\widetilde{x}, y) \\
&= \begin{cases} \text{the least } y \text{ such that } R(\widetilde{x}, y) \text{ holds}, & \text{if there is such a } y \\ \text{undefined}, & \text{otherwise.} \end{cases}
\end{aligned}
$$

is recursive.

Comment

The μ-operator allows one to define partial functions.

Comment

The μ-operator allows one to define partial functions.

The diagonalisation argument does not apply to the set $\mathfrak{F}$ of recursive functions.

Comment

The μ-operator allows one to define partial functions.

The diagonalisation argument does not apply to the set $\mathfrak{F}$ of recursive functions.

Using the μ-operator, one may define total functions that are not primitive recursive.

Definable Function

A function is definable if there is a recursive function calculating it.

Ackermann Function

The Ackermann function [1928] is defined as follows:

$$\begin{aligned} \psi(0, y) &\simeq y + 1, \\ \psi(x + 1, 0) &\simeq \psi(x, 1), \\ \psi(x + 1, y + 1) &\simeq \psi(x, \psi(x + 1, y)). \end{aligned}$$

The equations clearly define a total function.

Ackermann is not Primitive Recursive

Theorem

The Ackermann function grows faster than every primitive recursive function.

Ackermann Function is Recursive

Theorem

The Ackermann function is recursive.

Main Result

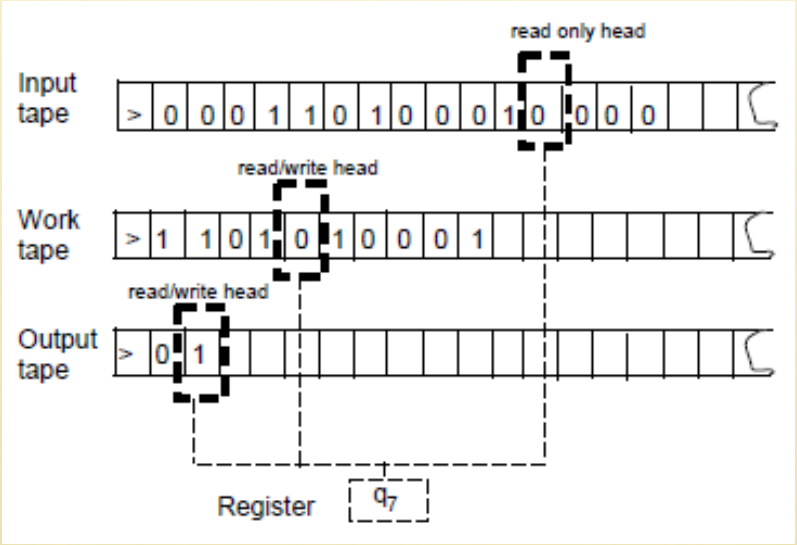
Theorem. All recursive functions are URM-definable.

Turing Machine

A k-tape Turing Machine $\mathbb{M}$ has k-tapes such that

- The first tape is the read-only input tape.
- The other $k-1$ tapes are the read/write work tapes.
- The k-th tape is also used as the output tape.

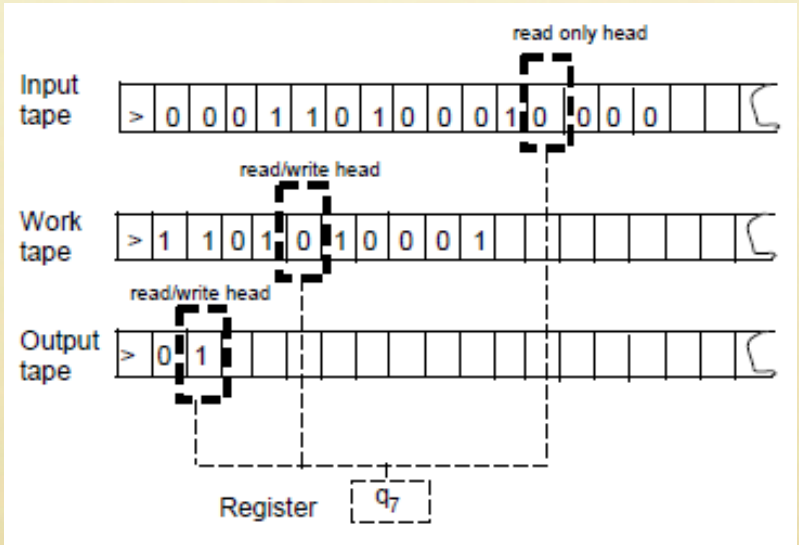
Every tape comes with a read/write head.



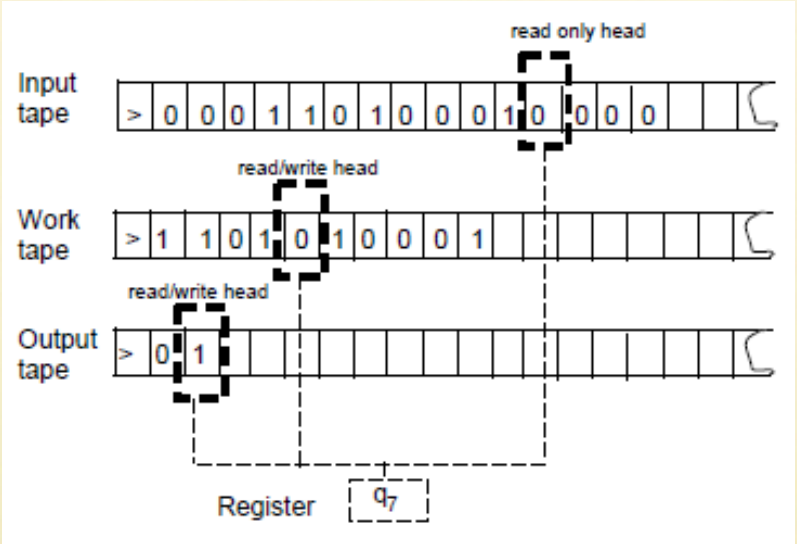
Turing Machine

The machine is described by a tuple (Γ, Q, δ) containing

- A finite set Γ, called alphabet, of symbols. It contains a blank symbol $\square$, a start symbol $\triangleright$, and the digits 0 and 1.
- A finite set Q of states. It contains a start state q_{s} and a halting state q_{h}.
- A transition function $\delta : Q \times \Gamma^k \to Q \times \Gamma^{k-1} \times \{\leftarrow, -, \rightarrow\}^k$, describing the rules of each computation step.



Computation and Configuration



Configuration, initial configuration, final configuration, computation step

Simulating TM by URM

Suppose $\mathbb{M}$ is a 3-tape TM with the alphabet $\{0, 1, \square, \triangleright\}$.

Simulating TM by URM

Suppose $\mathbb{M}$ is a 3-tape TM with the alphabet $\{0, 1, \square, \triangleright\}$.

The URM that simulates $\mathbb{M}$ can be designed as follows:

- Suppose that R_m is the right most register that is used by a program calculating $x \dot{-} 1$.
- The head positions are stored in $R_{m+1}, R_{m+2}, R_{m+3}$.
- The three binary strings in the tapes are stored respectively in
  $R_{m+4}, R_{m+7}, R_{m+10}, \ldots$,
  $R_{m+5}, R_{m+8}, R_{m+11}, \ldots$,
  $R_{m+6}, R_{m+9}, R_{m+12}, \ldots$.
- The states of $\mathbb{M}$ are encoded by the states of the URM.
- The transition function of $\mathbb{M}$ can be easily simulated by the program of the URM.

Fundamental Result

Theorem. The set of functions definable (the Turing Machine Model, the URM Model) is precisely the set of functions definable in the Recursive Function Model.

Church-Turing Thesis

Church-Turing Thesis.

The functions definable in all computation models are the same. They are precisely the computable functions.

Use of Church-Turing Thesis

Church-Turing Thesis allows us to give an informal argument for the computability of a function.

We will make use of a computable function without explicitly defining it.

Enumeration

An enumeration of a set X is a surjection $g : \mathbb{N} \to X$; this is often represented by writing $\{x_0, x_1, x_2, \ldots\}$.

It is an enumeration without repetition if g is injective.

Denumeration

A set X is denumerable if there is a bijection $f : X \to \mathbb{N}$.
(denumerate = denote + enumerate)

Denumeration

A set X is denumerable if there is a bijection $f : X \to \mathbb{N}$.
(denumerate = denote + enumerate)

Let X be a set of "finite objects".

Then X is effectively denumerable if there is a bijection $f : X \to \mathbb{N}$ such that both f and f^{-1} are computable.

Effective Denumerable Set

Fact. $\mathbb{N} \times \mathbb{N}$ is effectively denumerable.

Effective Denumerable Set

Fact. $\mathbb{N} \times \mathbb{N}$ is effectively denumerable.

Proof. A bijection $\pi : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ is defined by

$$\begin{aligned} \pi(m, n) &\stackrel{\text{def}}{=} 2^m(2n + 1) - 1, \\ \pi^{-1}(l) &\stackrel{\text{def}}{=} (\pi_1(l), \pi_2(l)), \end{aligned}$$

where

$$\begin{aligned} \pi_1(x) &\stackrel{\text{def}}{=} (x + 1)_1, \\ \pi_2(x) &\stackrel{\text{def}}{=} ((x + 1)/2^{\pi_1(x)} - 1)/2. \end{aligned}$$

Effective Denumerable Set

Fact. $\mathbb{N}^+ \times \mathbb{N}^+ \times \mathbb{N}^+$ is effectively denumerable.

Effective Denumerable Set

Fact. $\mathbb{N}^+ \times \mathbb{N}^+ \times \mathbb{N}^+$ is effectively denumerable.

Proof. A bijection $\zeta : \mathbb{N}^+ \times \mathbb{N}^+ \times \mathbb{N}^+ \to \mathbb{N}$ is defined by

$$\begin{aligned} \zeta(m,n,q) &\stackrel{\text{def}}{=} \pi(\pi(m-1,n-1),q-1), \\ \zeta^{-1}(l) &\stackrel{\text{def}}{=} (\pi_1(\pi_1(l))+1, \pi_2(\pi_1(l))+1, \pi_2(l)+1). \end{aligned}$$

Effective Denumerable Set

Fact. $\bigcup_{k>0} \mathbb{N}^k$ is effectively denumerable.

Proof. A bijection $\tau : \bigcup_{k>0} \mathbb{N}^k \to \mathbb{N}$ is defined by

$$\begin{aligned}\tau(a_1, \ldots, a_k) \stackrel{\text{def}}{=} & \; 2^{a_1} + 2^{a_1+a_2+1} + 2^{a_1+a_2+a_3+2} + \ldots \\ & + 2^{a_1+a_2+a_3+\ldots,a_k+k-1} - 1.\end{aligned}$$

Now given x it is easy to find $b_1 < b_2 < \ldots < b_k$ such that

$$2^{b_1} + 2^{b_2} + 2^{b_3} + \ldots + 2^{b_k} = x + 1.$$

It is then clear how to calculate $a_1, a_2, a_3, \ldots, a_k$. Details are next.

Encoding Program

Let $\mathcal{I}$ be the set of all instructions.

Let $\mathcal{P}$ be the set of all programs.

The objects in $\mathcal{I}$, and $\mathcal{P}$ as well, are “finite objects”.

Encoding Program

Theorem. $\mathcal{I}$ is effectively denumerable.

Proof. The bijection $\beta : \mathcal{I} \to \mathbb{N}$ is defined as follows:

$$\begin{aligned}
\beta(Z(n)) &= 4(n-1), \\
\beta(S(n)) &= 4(n-1)+1, \\
\beta(T(m,n)) &= 4\pi(m-1,n-1)+2, \\
\beta(J(m,n,q)) &= 4\zeta(m,n,q)+3.
\end{aligned}$$

The converse β^{-1} is easy.

Encoding Program

Theorem. $\mathcal{P}$ is effectively denumerable.

Proof. The bijection $\gamma : \mathcal{P} \to \mathbb{N}$ is defined as follows:

$$\gamma(P) \quad = \quad \tau(\beta(I_1), \ldots, \beta(I_s)),$$

assuming $P = I_1, \ldots, I_s$.

The converse γ^{-1} is obvious.

Gödel Number of Program

The value $\gamma(P)$ is called the Gödel number of P.

URM IS RECURSIVE

GÖDEL INDEX

Basic Idea

We see a number as an index for a problem/function if it is the Gödel number of a programme that solves/calculates the problem/function.

Definition

Suppose $a \in \mathbb{N}$ and $n \geq 1$.

$$\begin{aligned}
\phi_a^{(n)} &= \text{the } n \text{ ary function computed by } P_a \\
&= f_{P_a}^{(n)}, \\
W_a^{(n)} &= \text{the domain of } \phi_a^{(n)} = \{(x_1, \ldots, x_n) \mid P_a(x_1, \ldots, x_n) \downarrow\}, \\
E_a^{(n)} &= \text{the range of } \phi_a^{(n)}.
\end{aligned}$$

The super script (n) is omitted when $n = 1$.

Gödel Index for Computable Function

Suppose f is an n-ary computable function..

A number a is an index for f if $f = \phi_a^{(n)}$.

Diagonal Method

Suppose there is a sequence $f_0, f_1, \ldots, f_n, \ldots$

Diagonalize out of $f_0, f_1, \ldots$ by making f differ from f_n at n.

Enumeration of Computable Function

Proposition

$\mathcal{C}_n$, and $\mathcal{C}$ as well, is denumerable.

Enumeration of Computable Function

Proposition

$\mathcal{C}_n$, and $\mathcal{C}$ as well, is denumerable.

We may list for example all the elements of $\mathcal{C}_n$ as $\phi_0^{(n)}, \phi_1^{(n)}, \phi_2^{(n)}, \ldots$.

S-m-n Theorem, the Unary Case

Fact

Suppose that $f(x, y)$ is a computable function. There is a primitive recursive function $k(x)$ such that

$$f(x, y) \simeq \phi_{k(x)}(y).$$

S-m-n Theorem

S-m-n Theorem

For m, n, there is an injective primitive recursive $(m+1)$-function $s_n^m(x, \widetilde{x})$ such that for all e the following holds:

$$\phi_e^{(m+n)}(\widetilde{x}, \widetilde{y}) \simeq \phi_{s_n^m(e,\widetilde{x})}^{(n)}(\widetilde{y})$$

S-m-n Theorem

S-m-n Theorem

For m, n, there is an injective primitive recursive $(m+1)$-function $s_n^m(x, \widetilde{x})$ such that for all e the following holds:

$$\phi_e^{(m+n)}(\widetilde{x}, \widetilde{y}) \simeq \phi_{s_n^m(e,\widetilde{x})}^{(n)}(\widetilde{y})$$

S-m-n Theorem is also called **Parameter Theorem**.

General Remark

There are universal programs that embody all the programs.

A program is universal if upon receiving the Gödel number of a program it simulates the program indexed by the number.

Universal Function

The universal function for n-ary computable functions is the $(n+1)$-ary function $\psi_U^{(n)}$ defined by

$$\psi_U^{(n)}(e, x_1, \ldots, x_n) \simeq \phi_e^{(n)}(x_1, \ldots, x_n).$$

We write ψ_U for $\psi_U^{(1)}$.

Universal Function

The universal function for n-ary computable functions is the $(n+1)$-ary function $\psi_U^{(n)}$ defined by

$$\psi_U^{(n)}(e, x_1, \ldots, x_n) \simeq \phi_e^{(n)}(x_1, \ldots, x_n).$$

We write ψ_U for $\psi_U^{(1)}$.

Question: Is $\psi_U^{(n)}$ computable?

Enumeration Theorem

Enumeration Theorem

For each n, the universal function $\psi_U^{(n)}$ is computable.

Recursion Theorem

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 S-m-n Theorem there is an injective primitive recursive function $s(x)$ such that for all x

$$\phi_{s(x)}(y) \simeq \begin{cases} \phi_{\phi_x(x)}(y), & \text{if } \phi_x(x)\downarrow; \\ \uparrow, & \text{otherwise.} \end{cases}$$

Let v be such that $\phi_v = s \circ f$. Obviously ϕ_v is total and $\phi_v(v)\downarrow$.

$$\phi_{s(v)} = \phi_{\phi_v(v)} = \phi_{f(s(v))}$$

We are done by letting n be $s(v)$.

Decidability and Undecidability

A predicate $M(\mathbf{x})$ is decidable if its characteristic function $c_M(\mathbf{x})$ given by

$$c_M(\mathbf{x}) = \begin{cases} 1, & \text{if } M(\mathbf{x}) \text{ holds,} \\ 0, & \text{if } M(\mathbf{x}) \text{ does not hold.} \end{cases}$$

is computable.

Decidability and Undecidability

A predicate $M(\mathbf{x})$ is decidable if its characteristic function $c_M(\mathbf{x})$ given by

$$c_M(\mathbf{x}) = \begin{cases} 1, & \text{if } M(\mathbf{x}) \text{ holds,} \\ 0, & \text{if } M(\mathbf{x}) \text{ does not hold.} \end{cases}$$

is computable.

The predicate $M(\mathbf{x})$ is undecidable if it is not decidable.

Undecidability Result

Theorem

The problem '$x \in W_x$' is undecidable.

Many-One Reduction

The set A is many-one reducible, or m-reducible, to the set B if there is a total computable function f such that

$$x \in A \text{ iff } f(x) \in B$$

for all x. We shall write $A \leq_m B$ or more explicitly $f : A \leq_m B$.

If f is injective, then it is a one-one reducibility, denoted by $\leq_1$.

Many-One Reduction

- $\leq_m$ is reflexive and transitive.

Many-One Reduction

- $\leq_m$ is reflexive and transitive.
- $A \leq_m B$ iff $\overline{A} \leq_m \overline{B}$.

Many-One Reduction

- $\leq_m$ is reflexive and transitive.
- $A \leq_m B$ iff $\overline{A} \leq_m \overline{B}$.
- $A \leq_m \mathbb{N}$ iff $A = \mathbb{N}$; $A \leq_m \emptyset$ iff $A = \emptyset$.

Many-One Reduction

- $\leq_m$ is reflexive and transitive.
- $A \leq_m B$ iff $\overline{A} \leq_m \overline{B}$.
- $A \leq_m \mathbb{N}$ iff $A = \mathbb{N}$; $A \leq_m \emptyset$ iff $A = \emptyset$.
- $\mathbb{N} \leq_m A$ iff $A \neq \emptyset$; $\emptyset \leq_m A$ iff $A \neq \mathbb{N}$.

m-Degree

Two sets A, B are many-one equivalent, notation $A \equiv_m B$, if $A \leq_m B$ and $B \leq_m A$.

Similarly $A \equiv_1 B$ if $A \leq_1 B$ and $B \leq_1 A$.

Clearly both $\equiv_m$ and $\equiv_1$ an equivalence relation.

Let $d_m(A)$ be $\{B \mid A \equiv_m B\}$.

The class $d_m(A)$ is called the m-degree represented by A.

m-Degree

The set of m-degrees is ranged over by $\mathbf{a}, \mathbf{b}, \mathbf{c}, \ldots$.

$\mathbf{a} \leq_m \mathbf{b}$ iff $A \leq_m B$ for some $A \in \mathbf{a}$ and $B \in \mathbf{b}$.

$\mathbf{a} <_m \mathbf{b}$ iff $\mathbf{a} \leq_m \mathbf{b}$ and $\mathbf{b} \not\leq_m \mathbf{a}$.

The relation $\leq_m$ is a partial order.

The Structure of m-Degree

Proposition

The m-degrees form a distributive lattice.

Definition of Recursive Set

Let A be a subset of $\mathbb{N}$. The characteristic function of A is given by

$$c_A(x) = \begin{cases} 1, & \text{if } x \in A, \\ 0, & \text{if } x \notin A. \end{cases}$$

A is recursive if $c_A(x)$ is computable.

Fact about Recursive Set

Fact. If A is recursive then $\overline{A}$ is recursive.

Fact about Recursive Set

Fact. If A is recursive then $\overline{A}$ is recursive.

Fact. If A is recursive and $B \neq \emptyset, \mathbb{N}$, then $A \leq_m B$.

Fact about Recursive Set

Fact. If A is recursive then $\overline{A}$ is recursive.

Fact. If A is recursive and $B \neq \emptyset, \mathbb{N}$, then $A \leq_m B$.

Fact. If A, B are recursive and $A, B, \overline{A}, \overline{B}$ are infinite then $A \equiv B$.

Fact about Recursive Set

Fact. If A is recursive then $\overline{A}$ is recursive.

Fact. If A is recursive and $B \neq \emptyset, \mathbb{N}$, then $A \leq_m B$.

Fact. If A, B are recursive and $A, B, \overline{A}, \overline{B}$ are infinite then $A \equiv B$.

Fact. If $A \leq_m B$ and B is recursive, then A is recursive.

Fact about Recursive Set

Fact. If A is recursive then $\overline{A}$ is recursive.

Fact. If A is recursive and $B \neq \emptyset, \mathbb{N}$, then $A \leq_m B$.

Fact. If A, B are recursive and $A, B, \overline{A}, \overline{B}$ are infinite then $A \equiv B$.

Fact. If $A \leq_m B$ and B is recursive, then A is recursive.

Fact. If $A \leq_m B$ and A is not recursive, then B is not recursive.

A Characterization of Recursive Set

Theorem. An infinite set is recursive iff it is the range of a total increasing computable function.

Unsolvable Problem

A decision problem $f : \mathbb{N} \to \{0, 1\}$ is solvable if it is computable and $dom(f)$ is recursive.

It is unsolvable if it is not solvable.

Non-recursive $\Leftrightarrow$ Unsolvable $\Leftrightarrow$ Undecidable

Some Important Undecidable Sets

Here are some important undecidable sets:

$$
\begin{aligned}
K &= \{x \mid x \in W_x\}, \\
K_0 &= \{\pi(x,y) \mid x \in W_y\}, \\
K_1 &= \{x \mid W_x \neq \emptyset\}, \\
\mathit{Fin} &= \{x \mid W_x \text{ is finite}\}, \\
\mathit{Inf} &= \{x \mid W_x \text{ is infinite}\}, \\
\mathit{Con} &= \{x \mid \phi_x \text{ is total and constant}\}, \\
\mathit{Tot} &= \{x \mid \phi_x \text{ is total}\}, \\
\mathit{Cof} &= \{x \mid W_x \text{ is cofinite}\}, \\
\mathit{Rec} &= \{x \mid W_x \text{ is recursive}\}, \\
\mathit{Ext} &= \{x \mid \phi_x \text{ is extensible to a total recursive function}\}.
\end{aligned}
$$

Rice Theorem

Rice Theorem. (1953)

If $\emptyset \subsetneq \mathcal{B} \subsetneq \mathcal{C}$, then $\{x \mid \phi_x \in \mathcal{B}\}$ is not recursive.

The Definition of R.E. Set

The partial characteristic function of a set A is given by

$$\chi_A(x) = \begin{cases} 1, & \text{if } x \in A, \\ \uparrow, & \text{if } x \notin A. \end{cases}$$

A is recursively enumerable if χ_A is computable.

We shall often abbreviate 'recursively enumerable set' to 'r.e. set'.

Partially Decidable Problem

A problem $f : \mathbb{N} \to \{0, 1\}$ is partially decidable if $dom(f)$ is r.e.

Partially Decidable Predicate

A predicate $M(\widetilde{x})$ of natural number is partially decidable if its partial characteristic function

$$\chi_M(\widetilde{x}) = \begin{cases} 1, & \text{if } M(\widetilde{x}) \text{ holds,} \\ \uparrow, & \text{if } M(\widetilde{x}) \text{ does not hold,} \end{cases}$$

is computable.

$$\begin{aligned} \text{Partially Decidable Problem} &\Leftrightarrow \text{Partially Decidable Predicate} \\ &\Leftrightarrow \text{Recursively Enumerable Set} \end{aligned}$$

Example

The halting problem is partially decidable. Its partial characteristic function is given by

$$\chi_H(x, y) = \begin{cases} 1, & \text{if } P_x(y) \downarrow, \\ \uparrow, & \text{otherwise.} \end{cases}$$

Example

The halting problem is partially decidable. Its partial characteristic function is given by

$$\chi_H(x, y) = \begin{cases} 1, & \text{if } P_x(y) \downarrow, \\ \uparrow, & \text{otherwise.} \end{cases}$$

K, K_0, K_1 are r.e..

Example

The halting problem is partially decidable. Its partial characteristic function is given by

$$\chi_H(x,y) = \begin{cases} 1, & \text{if } P_x(y)\downarrow, \\ \uparrow, & \text{otherwise.} \end{cases}$$

K, K_0, K_1 are r.e.. But none of $\overline{K}, \overline{K_0}, \overline{K_1}$ is r.e..

Index for Recursively Enumerable Set

A set is r.e. iff it is the domain of a unary computable function.

Index for Recursively Enumerable Set

A set is r.e. iff it is the domain of a unary computable function.

So $W_0, W_1, W_2, \ldots$ is an enumeration of all r.e. sets.

Index for Recursively Enumerable Set

A set is r.e. iff it is the domain of a unary computable function.

So $W_0, W_1, W_2, \ldots$ is an enumeration of all r.e. sets.

Every r.e. set has an infinite number of indexes.

Uniformisation Theorem

Uniformisation Theorem. If $R(x, y)$ is partially decidable, then there is a computable function $c(x)$ such that $c(x) \downarrow$ iff $\exists y.R(x, y)$ and $c(x) \downarrow$ implies $R(x, c(x))$.

Uniformisation Theorem

Uniformisation Theorem. If $R(x, y)$ is partially decidable, then there is a computable function $c(x)$ such that $c(x) \downarrow$ iff $\exists y.R(x, y)$ and $c(x) \downarrow$ implies $R(x, c(x))$.

We may think of $c(x)$ as a choice function for $R(x, y)$. The theorem states that the choice function is computable.

A is r.e. iff there is a partially decidable predicate $R(x, y)$ such that $x \in A$ iff $\exists y.R(x, y)$.

Complementation Theorem

Complementation Theorem. A is recursive iff A and $\overline{A}$ are r.e.

Quantifier Contraction Theorem

Quantifier Contraction Theorem. If $M(\widetilde{x}, y)$ is partially decidable, so is $\exists y.M(\widetilde{x}, y)$.

Normal Form Theorem

Normal Form Theorem. $M(\widetilde{x})$ is partially decidable iff there is a primitive recursive predicate $R(\widetilde{x}, y)$ such that $M(\widetilde{x})$ iff $\exists y.R(\widetilde{x}, y)$.

Graph Theorem

Graph Theorem. Let $f(x)$ be a partial function. Then $f(x)$ is computable iff the predicate '$f(x) \simeq y$' is partially decidable iff $\{\pi(x, y) \mid f(x) \simeq y\}$ is r.e.

Listing Theorem

Listing Theorem. A is r.e. iff either $A = \emptyset$ or A is the range of a unary total computable function.