Introduction to Theory of Computability

Nikolaj Popov and Tudor Jebelean

Research Institute for Symbolic Computation, Linz

{popov,jebelean}@risc.uni-linz.ac.at

Outline

Outline

Introduction

Various notions of computation developed by Gödel, Church, Turing and Kleene

The three computational models (recursion, λ-calculus, and Turing machine) were shown to be equivalent (1934).

Church-Turing thesis

Any real-world computation can be translated into an equivalent computation involving a Turing machine (or a program in any *reasonable* programming language).

The intuitive notion of effective computability for functions and algorithms is formally expressed by Turing machines or the lambda calculus.

A function is computable, in the intuitive sense, if and only if it is Turing-computable.

Introduction

Various notions of computation developed by Gödel, Church, Turing and Kleene

The three computational models (recursion, λ-calculus, and Turing machine) were shown to be equivalent (1934).

Church-Turing thesis

Any real-world computation can be translated into an equivalent computation involving a Turing machine (or a program in any *reasonable* programming language).

The intuitive notion of effective computability for functions and algorithms is formally expressed by Turing machines or the lambda calculus.

A function is computable, in the intuitive sense, if and only if it is Turing-computable.

Introduction

Various notions of computation developed by Gödel, Church, Turing and Kleene

The three computational models (recursion, λ-calculus, and Turing machine) were shown to be equivalent (1934).

Church-Turing thesis

Any real-world computation can be translated into an equivalent computation involving a Turing machine (or a program in any *reasonable* programming language).

The intuitive notion of effective computability for functions and algorithms is formally expressed by Turing machines or the lambda calculus.

A function is computable, in the intuitive sense, if and only if it is Turing-computable.

Introduction

Various notions of computation developed by Gödel, Church, Turing and Kleene

The three computational models (recursion, λ-calculus, and Turing machine) were shown to be equivalent (1934).

Church-Turing thesis

Any real-world computation can be translated into an equivalent computation involving a Turing machine (or a program in any *reasonable* programming language).

The intuitive notion of effective computability for functions and algorithms is formally expressed by Turing machines or the lambda calculus.

A function is computable, in the intuitive sense, if and only if it is Turing-computable.

Introduction

Various notions of computation developed by Gödel, Church, Turing and Kleene

The three computational models (recursion, λ-calculus, and Turing machine) were shown to be equivalent (1934).

Church-Turing thesis

Any real-world computation can be translated into an equivalent computation involving a Turing machine (or a program in any *reasonable* programming language).

The intuitive notion of effective computability for functions and algorithms is formally expressed by Turing machines or the lambda calculus.

A function is computable, in the intuitive sense, if and only if it is Turing-computable.

Mathematical Preliminaries

Natural Numbers

$\mathbb{N} = \{0, 1, \ldots\}$

Sets

$\{a_1, a_2, \ldots, a_n\}$ the order of the elements is irrelevant

n-tuples

$(a_1, a_2, \ldots, a_n) = (b_1, b_2, \ldots, b_n)$
iff
$a_1 = b_1, \ldots, a_n = b_n$

Operations on Sets

$A \cup B = \{\overline{a} \mid \overline{a} \in A \text{ or } \overline{a} \in B\}$
$A \cap B = \{\overline{a} \mid \overline{a} \in A \text{ and } \overline{a} \in B\}$
$A \setminus B = \{\overline{a} \mid \overline{a} \in A \text{ and } \overline{a} \notin B\}$
$\overline{A} = \mathbb{N}^n \setminus A$

Mathematical Preliminaries

Natural Numbers
$\mathbb{N} = \{0, 1, \dots\}$

Sets
$\{a_1, a_2, \dots, a_n\}$ the order of the elements is irrelevant

n-tuples
$(a_1, a_2, \dots, a_n) = (b_1, b_2, \dots, b_n)$
iff
$a_1 = b_1, \dots, a_n = b_n$

Operations on Sets
$A \cup B = \{\overline{a} \mid \overline{a} \in A \text{ or } \overline{a} \in B\}$
$A \cap B = \{\overline{a} \mid \overline{a} \in A \text{ and } \overline{a} \in B\}$
$A \setminus B = \{\overline{a} \mid \overline{a} \in A \text{ and } \overline{a} \notin B\}$
$\overline{A} = \mathbb{N}^n \setminus A$

Mathematical Preliminaries

Natural Numbers

$\mathbb{N} = \{0, 1, \ldots\}$

Sets

$\{a_1, a_2, \ldots, a_n\}$ the order of the elements is irrelevant

n-tuples

$(a_1, a_2, \ldots, a_n) = (b_1, b_2, \ldots, b_n)$
iff
$a_1 = b_1, \ldots, a_n = b_n$

Operations on Sets

$A \cup B = \{\overline{a} \mid \overline{a} \in A \text{ or } \overline{a} \in B\}$
$A \cap B = \{\overline{a} \mid \overline{a} \in A \text{ and } \overline{a} \in B\}$
$A \setminus B = \{\overline{a} \mid \overline{a} \in A \text{ and } \overline{a} \notin B\}$
$\overline{A} = \mathbb{N}^n \setminus A$

Mathematical Preliminaries

Natural Numbers

$\mathbb{N} = \{0, 1, \dots\}$

Sets

$\{a_1, a_2, \dots, a_n\}$ the order of the elements is irrelevant

n-tuples

$(a_1, a_2, \dots, a_n) = (b_1, b_2, \dots, b_n)$
iff
$a_1 = b_1, \dots, a_n = b_n$

Operations on Sets

$A \cup B = \{\overline{a} \mid \overline{a} \in A \text{ or } \overline{a} \in B\}$
$A \cap B = \{\overline{a} \mid \overline{a} \in A \text{ and } \overline{a} \in B\}$
$A \setminus B = \{\overline{a} \mid \overline{a} \in A \text{ and } \overline{a} \notin B\}$
$\overline{A} = \mathbb{N}^n \setminus A$

Mathematical Preliminaries

Domain of a function
$Dom[f] = \{\overline{x} \mid f[\overline{x}] \textit{ is defined}\}$

Range of a function
$Ran[f] = \{y \mid \exists \overline{x} \in Dom[f] \wedge f[\overline{x}] = y\}$

Graph of a function
$Graph[f] = \{(\overline{x}, y) \mid \exists \overline{x} \in Dom[f] \wedge f[\overline{x}] = y\}$

Partial equality
$f[\overline{x}] \simeq y \quad \Leftrightarrow \quad (\overline{x}, y) \in Graph[f]$

Mathematical Preliminaries

Domain of a function

$Dom[f] = \{\overline{x} \mid f[\overline{x}] \text{ is defined}\}$

Range of a function

$Ran[f] = \{y \mid \exists \overline{x} \in Dom[f] \wedge f[\overline{x}] = y\}$

Graph of a function

$Graph[f] = \{(\overline{x}, y) \mid \exists \overline{x} \in Dom[f] \wedge f[\overline{x}] = y\}$

Partial equality

$f[\overline{x}] \simeq y \quad \Leftrightarrow \quad (\overline{x}, y) \in Graph[f]$

Mathematical Preliminaries

Domain of a function

$Dom[f] = \{\overline{x} \mid f[\overline{x}] \textit{ is defined}\}$

Range of a function

$Ran[f] = \{y \mid \exists \overline{x} \in Dom[f] \wedge f[\overline{x}] = y\}$

Graph of a function

$Graph[f] = \{(\overline{x}, y) \mid \exists \overline{x} \in Dom[f] \wedge f[\overline{x}] = y\}$

Partial equality

$f[\overline{x}] \simeq y \quad \Leftrightarrow \quad (\overline{x}, y) \in Graph[f]$

Mathematical Preliminaries

Domain of a function
$Dom[f] = \{\overline{x} \mid f[\overline{x}] \textit{ is defined}\}$

Range of a function
$Ran[f] = \{y \mid \exists \overline{x} \in Dom[f] \wedge f[\overline{x}] = y\}$

Graph of a function
$Graph[f] = \{(\overline{x}, y) \mid \exists \overline{x} \in Dom[f] \wedge f[\overline{x}] = y\}$

Partial equality
$f[\overline{x}] \simeq y \quad \Leftrightarrow \quad (\overline{x}, y) \in Graph[f]$

Mathematical Preliminaries

f is defined

$f[\overline{x}]\downarrow \;\Leftrightarrow\; (\overline{x}, y) \in \mathit{Graph}[f]$

Partial equality $\simeq$

$f[\overline{x}] \simeq g[\overline{x}]$

iff

$f[\overline{x}]\downarrow \;\Leftrightarrow\; g[\overline{x}]\downarrow$

$f[\overline{x}]\downarrow \;\Rightarrow\; f[\overline{x}] = g[\overline{x}]$

f is computable

f is computable function iff
there exists a program P which computes it

Mathematical Preliminaries

f is defined

$f[\overline{x}]\downarrow \;\Leftrightarrow\; (\overline{x}, y) \in Graph[f]$

Partial equality $\simeq$

$f[\overline{x}] \simeq g[\overline{x}]$

iff

$f[\overline{x}]\downarrow \;\Leftrightarrow\; g[\overline{x}]\downarrow$

$f[\overline{x}]\downarrow \;\Rightarrow\; f[\overline{x}] = g[\overline{x}]$

f is computable

f is computable function iff

there exists a program P which computes it

Mathematical Preliminaries

f is defined

$f[\overline{x}]\downarrow \;\Leftrightarrow\; (\overline{x}, y) \in Graph[f]$

Partial equality $\simeq$

$f[\overline{x}] \simeq g[\overline{x}]$
iff
$f[\overline{x}]\downarrow \;\Leftrightarrow\; g[\overline{x}]\downarrow$
$f[\overline{x}]\downarrow \;\Rightarrow\; f[\overline{x}] = g[\overline{x}]$

f is computable

f is computable function iff
there exists a program P which computes it

Outline

Superposition

h is a superposition of $f, g_1, \ldots, g_k$

$h[\overline{x}] \simeq f[g_1[\overline{x}], \ldots, g_k[\overline{x}]]$

Theorem

Given the computable functions $f, g_1, \ldots, g_k$, then $h[\overline{x}] \simeq f[g_1[\overline{x}], \ldots, g_k[\overline{x}]]$ is computable function.

Superposition preserves computability.

Superposition

h is a superposition of $f, g_1, \ldots, g_k$

$h[\overline{x}] \simeq f[g_1[\overline{x}], \ldots, g_k[\overline{x}]]$

Theorem

Given the computable functions $f, g_1, \ldots, g_k$, then $h[\overline{x}] \simeq f[g_1[\overline{x}], \ldots, g_k[\overline{x}]]$ is computable function.

Superposition preserves computability.

Superposition

h is a superposition of $f, g_1, \ldots, g_k$

$h[\overline{x}] \simeq f[g_1[\overline{x}], \ldots, g_k[\overline{x}]]$

Theorem

Given the computable functions $f, g_1, \ldots, g_k$, then $h[\overline{x}] \simeq f[g_1[\overline{x}], \ldots, g_k[\overline{x}]]$ is computable function.

Superposition preserves computability.

Primitive Recursion

h is obtained by *weak* primitive recursion from g and a

$$h[x] \simeq \begin{cases} a & \Leftarrow \ x = 0 \\ g[x-1, h[x-1]] & \Leftarrow \ \text{o.w.} \end{cases}$$

h is obtained by primitive recursion from f and g

$$h[\overline{x}, y] \simeq \begin{cases} f[\overline{x}] & \Leftarrow \ y = 0 \\ g[\overline{x}, y-1, h[\overline{x}, y-1]] & \Leftarrow \ \text{o.w.} \end{cases}$$

Primitive recursion preserves computability.

Primitive Recursion

h **is obtained by *weak* primitive recursion from** g **and** a

$$h[x] \simeq \begin{cases} a & \Leftarrow \; x = 0 \\ g[x-1, h[x-1]] & \Leftarrow \; \text{o.w.} \end{cases}$$

h **is obtained by primitive recursion from** f **and** g

$$h[\overline{x}, y] \simeq \begin{cases} f[\overline{x}] & \Leftarrow \; y = 0 \\ g[\overline{x}, y-1, h[\overline{x}, y-1]] & \Leftarrow \; \text{o.w.} \end{cases}$$

Primitive recursion preserves computability.

Primitive Recursion

h is obtained by *weak* primitive recursion from g and a

$$h[x] \simeq \begin{cases} a & \Leftarrow \; x = 0 \\ g[x-1, h[x-1]] & \Leftarrow \; \text{o.w.} \end{cases}$$

h is obtained by primitive recursion from f and g

$$h[\overline{x}, y] \simeq \begin{cases} f[\overline{x}] & \Leftarrow \; y = 0 \\ g[\overline{x}, y-1, h[\overline{x}, y-1]] & \Leftarrow \; \text{o.w.} \end{cases}$$

Primitive recursion preserves computability.

Primitive Recursive Functions

The basic functions are primitive recursive

$O[x] \simeq 0$

$S[x] \simeq x + 1$

$I_i^n[\overline{x}] \simeq x_i$

The superposition is primitive recursive

If $f, g_1, \ldots, g_k$ are primitive recursive, then

$h[\overline{x}] \simeq f[g_1[\overline{x}], \ldots, g_k[\overline{x}]]$

is primitive recursive.

The primitive recursion is primitive recursive

If f and g are primitive recursive, then

$$h[\overline{x}, y] \simeq \begin{cases} f[\overline{x}] & \Leftarrow \ y = 0 \\ g[\overline{x}, y-1, h[\overline{x}, y-1]] & \Leftarrow \ \text{o.w.} \end{cases}$$

is primitive recursive.

Primitive Recursive Functions

The basic functions are primitive recursive

$O[x] \simeq 0$

$S[x] \simeq x + 1$

$I_i^n[\overline{x}] \simeq x_i$

The superposition is primitive recursive

If $f, g_1, \ldots, g_k$ are primitive recursive, then

$h[\overline{x}] \simeq f[g_1[\overline{x}], \ldots, g_k[\overline{x}]]$

is primitive recursive.

The primitive recursion is primitive recursive

If f and g are primitive recursive, then

$$h[\overline{x}, y] \simeq \begin{cases} f[\overline{x}] & \Leftarrow y = 0 \\ g[\overline{x}, y-1, h[\overline{x}, y-1]] & \Leftarrow \text{o.w.} \end{cases}$$

is primitive recursive.

Primitive Recursive Functions

The basic functions are primitive recursive

$O[x] \simeq 0$

$S[x] \simeq x + 1$

$I_i^n[\overline{x}] \simeq x_i$

The superposition is primitive recursive

If $f, g_1, \ldots, g_k$ are primitive recursive, then

$h[\overline{x}] \simeq f[g_1[\overline{x}], \ldots, g_k[\overline{x}]]$

is primitive recursive.

The primitive recursion is primitive recursive

If f and g are primitive recursive, then

$$h[\overline{x}, y] \simeq \begin{cases} f[\overline{x}] & \Leftarrow \; y = 0 \\ g[\overline{x}, y-1, h[\overline{x}, y-1]] & \Leftarrow \; \text{o.w.} \end{cases}$$

is primitive recursive.

Primitive Recursion

Theorem

All the primitive recursive functions are computable.

Theorem

All the primitive recursive functions are total.

Primitive Recursion

Theorem
All the primitive recursive functions are computable.

Theorem
All the primitive recursive functions are total.

Examples

Addition is primitive recursive

$f_1[x, y] \simeq x + y$

$$f_1[x, y] \simeq \begin{cases} x & \Leftarrow \; y = 0 \\ S[f_1[x, y-1]] & \Leftarrow \; \text{o.w.} \end{cases}$$

Multiplication is primitive recursive

$f_2[x, y] \simeq x.y$

$$f_2[x, y] \simeq \begin{cases} 0 & \Leftarrow \; y = 0 \\ x + f_2[x, y-1] & \Leftarrow \; \text{o.w.} \end{cases}$$

Examples

Addition is primitive recursive

$f_1[x, y] \simeq x + y$

$$f_1[x, y] \simeq \begin{cases} x & \Leftarrow \; y = 0 \\ S[f_1[x, y-1]] & \Leftarrow \; \text{o.w.} \end{cases}$$

Multiplication is primitive recursive

$f_2[x, y] \simeq x.y$

$$f_2[x, y] \simeq \begin{cases} 0 & \Leftarrow \; y = 0 \\ x + f_2[x, y-1] & \Leftarrow \; \text{o.w.} \end{cases}$$

Examples

Addition is primitive recursive

$f_1[x, y] \simeq x + y$

$$f_1[x, y] \simeq \begin{cases} x & \Leftarrow \; y = 0 \\ S[f_1[x, y - 1]] & \Leftarrow \; \text{o.w.} \end{cases}$$

Multiplication is primitive recursive

$f_2[x, y] \simeq x.y$

$$f_2[x, y] \simeq \begin{cases} 0 & \Leftarrow \; y = 0 \\ x + f_2[x, y - 1] & \Leftarrow \; \text{o.w.} \end{cases}$$

Examples

Addition is primitive recursive

$f_1[x, y] \simeq x + y$

$$f_1[x, y] \simeq \begin{cases} x & \Leftarrow \; y = 0 \\ S[f_1[x, y-1]] & \Leftarrow \; \text{o.w.} \end{cases}$$

Multiplication is primitive recursive

$f_2[x, y] \simeq x.y$

$$f_2[x, y] \simeq \begin{cases} 0 & \Leftarrow \; y = 0 \\ x + f_2[x, y-1] & \Leftarrow \; \text{o.w.} \end{cases}$$

Examples

Power is primitive recursive

$f_3[x, y] \simeq x^y$

$$f_3[x, y] \simeq \begin{cases} 1 & \Leftarrow \; y = 0 \\ x.f_3[x, y-1]] & \Leftarrow \; \text{o.w.} \end{cases}$$

Subtraction-dot-one is primitive recursive

$$f_4[x] \simeq x \dot{-} 1 \simeq \begin{cases} 0 & \Leftarrow \; x = 0 \\ x - 1 & \Leftarrow \; \text{o.w.} \end{cases}$$

$$f_4[x] \simeq \begin{cases} 0 & \Leftarrow \; x = 0 \\ I_1^2[x-1, f_4[x-1]] & \Leftarrow \; \text{o.w.} \end{cases}$$

Examples

Power is primitive recursive

$f_3[x, y] \simeq x^y$

$$f_3[x, y] \simeq \begin{cases} 1 & \Leftarrow \; y = 0 \\ x.f_3[x, y-1]] & \Leftarrow \; \text{o.w.} \end{cases}$$

Subtraction-dot-one is primitive recursive

$$f_4[x] \simeq x \dot{-} 1 \simeq \begin{cases} 0 & \Leftarrow \; x = 0 \\ x - 1 & \Leftarrow \; \text{o.w.} \end{cases}$$

$$f_4[x] \simeq \begin{cases} 0 & \Leftarrow \; x = 0 \\ I^2_1[x-1, f_4[x-1]] & \Leftarrow \; \text{o.w.} \end{cases}$$

Examples

Power is primitive recursive

$f_3[x, y] \simeq x^y$

$$f_3[x, y] \simeq \begin{cases} 1 & \Leftarrow \ y = 0 \\ x.f_3[x, y-1]] & \Leftarrow \ \text{o.w.} \end{cases}$$

Subtraction-dot-one is primitive recursive

$$f_4[x] \simeq x \dot{-} 1 \simeq \begin{cases} 0 & \Leftarrow \ x = 0 \\ x - 1 & \Leftarrow \ \text{o.w.} \end{cases}$$

$$f_4[x] \simeq \begin{cases} 0 & \Leftarrow \ x = 0 \\ I^2_1[x-1, f_4[x-1]] & \Leftarrow \ \text{o.w.} \end{cases}$$

Examples

Power is primitive recursive

$f_3[x, y] \simeq x^y$

$$f_3[x, y] \simeq \begin{cases} 1 & \Leftarrow \; y = 0 \\ x.f_3[x, y-1]] & \Leftarrow \; \text{o.w.} \end{cases}$$

Subtraction-dot-one is primitive recursive

$$f_4[x] \simeq x \dot{-} 1 \simeq \begin{cases} 0 & \Leftarrow \; x = 0 \\ x - 1 & \Leftarrow \; \text{o.w.} \end{cases}$$

$$f_4[x] \simeq \begin{cases} 0 & \Leftarrow \; x = 0 \\ I^2_1[x-1, f_4[x-1]] & \Leftarrow \; \text{o.w.} \end{cases}$$

Examples

Subtraction-dot is primitive recursive

$$f_5[x, y] \simeq x \dot{-} y \simeq \begin{cases} 0 & \Leftarrow \; x < y \\ x - y & \Leftarrow \; \text{o.w.} \end{cases}$$

$$f_5[x, y] \simeq \begin{cases} x & \Leftarrow \; y = 0 \\ f_5[x, y-1] \dot{-} 1 & \Leftarrow \; \text{o.w.} \end{cases}$$

Factorial is primitive recursive

$f_6[x] \simeq x!$

$$f_6[x] \simeq \begin{cases} 1 & \Leftarrow \; x = 0 \\ x . f_6[x-1]] & \Leftarrow \; \text{o.w.} \end{cases}$$

Examples

Subtraction-dot is primitive recursive

$$f_5[x, y] \simeq x \dot{-} y \simeq \begin{cases} 0 & \Leftarrow \; x < y \\ x - y & \Leftarrow \; \text{o.w.} \end{cases}$$

$$f_5[x, y] \simeq \begin{cases} x & \Leftarrow \; y = 0 \\ f_5[x, y - 1] \dot{-} 1 & \Leftarrow \; \text{o.w.} \end{cases}$$

Factorial is primitive recursive

$f_6[x] \simeq x!$

$$f_6[x] \simeq \begin{cases} 1 & \Leftarrow \; x = 0 \\ x.f_6[x - 1]] & \Leftarrow \; \text{o.w.} \end{cases}$$

Examples

Subtraction-dot is primitive recursive

$$f_5[x, y] \simeq x \dot{-} y \simeq \begin{cases} 0 & \Leftarrow \; x < y \\ x - y & \Leftarrow \; \text{o.w.} \end{cases}$$

$$f_5[x, y] \simeq \begin{cases} x & \Leftarrow \; y = 0 \\ f_5[x, y-1] \dot{-} 1 & \Leftarrow \; \text{o.w.} \end{cases}$$

Factorial is primitive recursive

$f_6[x] \simeq x!$

$$f_6[x] \simeq \begin{cases} 1 & \Leftarrow \; x = 0 \\ x.f_6[x-1]] & \Leftarrow \; \text{o.w.} \end{cases}$$

Examples

Subtraction-dot is primitive recursive

$$f_5[x,y] \simeq x \dot{-} y \simeq \begin{cases} 0 & \Leftarrow \; x < y \\ x - y & \Leftarrow \; \text{o.w.} \end{cases}$$

$$f_5[x,y] \simeq \begin{cases} x & \Leftarrow \; y = 0 \\ f_5[x, y-1] \dot{-} 1 & \Leftarrow \; \text{o.w.} \end{cases}$$

Factorial is primitive recursive

$f_6[x] \simeq x!$

$$f_6[x] \simeq \begin{cases} 1 & \Leftarrow \; x = 0 \\ x.f_6[x-1]] & \Leftarrow \; \text{o.w.} \end{cases}$$

Examples

Sign is primitive recursive

$$sg[x] \simeq \begin{cases} 0 & \Leftarrow \; x = 0 \\ 1 & \Leftarrow \; \text{o.w.} \end{cases}$$

$$sg[x] \simeq \begin{cases} 0 & \Leftarrow \; x = 0 \\ O[sg[x-1]] + 1 & \Leftarrow \; \text{o.w.} \end{cases}$$

Opposite-sign is primitive recursive

$$\overline{sg}[x] \simeq \begin{cases} 1 & \Leftarrow \; x = 0 \\ 0 & \Leftarrow \; \text{o.w.} \end{cases}$$

$$\overline{sg}[x] \simeq \begin{cases} 1 & \Leftarrow \; x = 0 \\ O[\overline{sg}[x-1]] & \Leftarrow \; \text{o.w.} \end{cases}$$

Examples

Sign is primitive recursive

$$sg[x] \simeq \begin{cases} 0 & \Leftarrow \; x = 0 \\ 1 & \Leftarrow \; \text{o.w.} \end{cases}$$

$$sg[x] \simeq \begin{cases} 0 & \Leftarrow \; x = 0 \\ O[sg[x-1]] + 1 & \Leftarrow \; \text{o.w.} \end{cases}$$

Opposite-sign is primitive recursive

$$\overline{sg}[x] \simeq \begin{cases} 1 & \Leftarrow \; x = 0 \\ 0 & \Leftarrow \; \text{o.w.} \end{cases}$$

$$\overline{sg}[x] \simeq \begin{cases} 1 & \Leftarrow \; x = 0 \\ O[\overline{sg}[x-1]] & \Leftarrow \; \text{o.w.} \end{cases}$$

Examples

Sign is primitive recursive

$$sg[x] \simeq \begin{cases} 0 & \Leftarrow \ x = 0 \\ 1 & \Leftarrow \ \text{o.w.} \end{cases}$$

$$sg[x] \simeq \begin{cases} 0 & \Leftarrow \ x = 0 \\ O[sg[x-1]] + 1 & \Leftarrow \ \text{o.w.} \end{cases}$$

Opposite-sign is primitive recursive

$$\overline{sg}[x] \simeq \begin{cases} 1 & \Leftarrow \ x = 0 \\ 0 & \Leftarrow \ \text{o.w.} \end{cases}$$

$$\overline{sg}[x] \simeq \begin{cases} 1 & \Leftarrow \ x = 0 \\ O[\overline{sg}[x-1]] & \Leftarrow \ \text{o.w.} \end{cases}$$

Examples

Sign is primitive recursive

$$sg[x] \simeq \begin{cases} 0 & \Leftarrow \ x = 0 \\ 1 & \Leftarrow \ \text{o.w.} \end{cases}$$

$$sg[x] \simeq \begin{cases} 0 & \Leftarrow \ x = 0 \\ O[sg[x-1]] + 1 & \Leftarrow \ \text{o.w.} \end{cases}$$

Opposite-sign is primitive recursive

$$\overline{sg}[x] \simeq \begin{cases} 1 & \Leftarrow \ x = 0 \\ 0 & \Leftarrow \ \text{o.w.} \end{cases}$$

$$\overline{sg}[x] \simeq \begin{cases} 1 & \Leftarrow \ x = 0 \\ O[\overline{sg}[x-1]] & \Leftarrow \ \text{o.w.} \end{cases}$$

Examples

Absolute value is primitive recursive

$mod[x, y] \simeq |x - y|$

$mod[x, y] \simeq (x \dot{-} y) + (y \dot{-} x)$

Minimum is primitive recursive

$min[x, y]$

$min[x, y] \simeq x \dot{-} (x \dot{-} y)$

Maximum is primitive recursive

$max[x, y]$

$max[x, y] \simeq x + (y \dot{-} x)$

Examples

Absolute value is primitive recursive

$$mod[x, y] \simeq |x - y|$$

$$mod[x, y] \simeq (x \dot{-} y) + (y \dot{-} x)$$

Minimum is primitive recursive

$min[x, y]$

$$min[x, y] \simeq x \dot{-} (x \dot{-} y)$$

Maximum is primitive recursive

$max[x, y]$

$$max[x, y] \simeq x + (y \dot{-} x)$$

Examples

Absolute value is primitive recursive

$mod[x, y] \simeq |x - y|$

$mod[x, y] \simeq (x \dot{-} y) + (y \dot{-} x)$

Minimum is primitive recursive

$min[x, y]$

$min[x, y] \simeq x \dot{-} (x \dot{-} y)$

Maximum is primitive recursive

$max[x, y]$

$max[x, y] \simeq x + (y \dot{-} x)$

Examples

Absolute value is primitive recursive

$mod[x, y] \simeq |x - y|$

$mod[x, y] \simeq (x \dot{-} y) + (y \dot{-} x)$

Minimum is primitive recursive

$min[x, y]$

$min[x, y] \simeq x \dot{-} (x \dot{-} y)$

Maximum is primitive recursive

$max[x, y]$

$max[x, y] \simeq x + (y \dot{-} x)$

Examples

Absolute value is primitive recursive

$mod[x, y] \simeq |x - y|$

$mod[x, y] \simeq (x \dot{-} y) + (y \dot{-} x)$

Minimum is primitive recursive

$min[x, y]$

$min[x, y] \simeq x \dot{-} (x \dot{-} y)$

Maximum is primitive recursive

$max[x, y]$

$max[x, y] \simeq x + (y \dot{-} x)$

Examples

Absolute value is primitive recursive

$mod[x, y] \simeq |x - y|$

$$mod[x, y] \simeq (x \dot{-} y) + (y \dot{-} x)$$

Minimum is primitive recursive

$min[x, y]$

$$min[x, y] \simeq x \dot{-} (x \dot{-} y)$$

Maximum is primitive recursive

$max[x, y]$

$$max[x, y] \simeq x + (y \dot{-} x)$$

Primitive recursion. Properties

Theorem *If then else*
Let f_0, f_1, g be primitive recursive.
Then

$$h[\overline{x}] \simeq \begin{cases} f_0[\overline{x}] & \Leftarrow \ g[\overline{x}] = 0 \\ f_1[\overline{x}] & \Leftarrow \ \text{o.w.} \end{cases}$$

is primitive recursive.

proof:
$h[\overline{x}] \simeq \overline{sg}[g[\overline{x}]].f_0[x] + sg[g[\overline{x}]].f_1[x]$

Primitive recursion. Properties

Theorem *If then else*

Let f_0, f_1, g be primitive recursive.
Then

$$h[\overline{x}] \simeq \begin{cases} f_0[\overline{x}] & \Leftarrow \; g[\overline{x}] = 0 \\ f_1[\overline{x}] & \Leftarrow \; \text{o.w.} \end{cases}$$

is primitive recursive.

proof:

$h[\overline{x}] \simeq \overline{sg}[g[\overline{x}]].f_0[x] + sg[g[\overline{x}]].f_1[x]$

Primitive recursion. Properties

Theorem *If then$_1$ ... then$_k$ else*

Let $f_0, \ldots, f_k, g_0, \ldots, g_{k-1}$ be primitive recursive.
Then

$$h[\overline{x}] \simeq \begin{cases} f_0[\overline{x}] & \Leftarrow \; g_0[\overline{x}] = 0 \\ f_1[\overline{x}] & \Leftarrow \; g_0[\overline{x}] \neq 0 \wedge g_1[\overline{x}] = 0 \\ \ldots & \\ \ldots & \\ f_k[\overline{x}] & \Leftarrow \; \text{o.w.} \end{cases}$$

is primitive recursive.

Partial Functions

While loop

input$[x]$
$y := 0$
while $f[x, y] > 0$ *do* $y := y + 1$
return$[y]$

g is obtained by minimization from f

$g[\overline{x}] \simeq y$
iff
$\forall z < y(f[\overline{x}, z] \downarrow \wedge f[\overline{x}, z] > 0)$
$f[\overline{x}, y] \simeq 0$

g is obtained by minimization from f

$g[\overline{x}] \simeq \mu y[f[\overline{x}, y] = 0]$

Partial Functions

While loop

$input[x]$
$y := 0$
$while \;\; f[x,y] > 0 \;\; do \;\; y := y + 1$
$return[y]$

g is obtained by minimization from f

$g[\overline{x}] \simeq y$

iff

$\forall z < y(f[\overline{x}, z]\downarrow \;\wedge\; f[\overline{x}, z] > 0)$

$f[\overline{x}, y] \simeq 0$

g is obtained by minimization from f

$g[\overline{x}] \simeq \mu y[f[\overline{x}, y] = 0]$

Partial Functions

While loop

$input[x]$
$y := 0$
$while \;\; f[x,y] > 0 \;\; do \;\; y := y+1$
$return[y]$

g is obtained by minimization from f

$g[\overline{x}] \simeq y$
iff
$\forall z < y(f[\overline{x},z]\downarrow \; \wedge f[\overline{x},z] > 0)$
$f[\overline{x},y] \simeq 0$

g is obtained by minimization from f

$g[\overline{x}] \simeq \mu y[f[\overline{x},y] = 0]$

Partial Functions

The basic functions are partial

$O[x] \simeq 0$

$S[x] \simeq x + 1$

$I_i^n[\overline{x}] \simeq x_i$

The superposition is partial

If $f, g_1, \ldots, g_k$ are partial, then

$h[\overline{x}] \simeq f[g_1[\overline{x}], \ldots, g_k[\overline{x}]]$

is partial.

Partial Functions

The basic functions are partial

$O[x] \simeq 0$

$S[x] \simeq x + 1$

$I_i^n[\overline{x}] \simeq x_i$

The superposition is partial

If $f, g_1, \ldots, g_k$ are partial, then

$h[\overline{x}] \simeq f[g_1[\overline{x}], \ldots, g_k[\overline{x}]]$

is partial.

Partial Functions

The primitive recursion is partial

If f and g are partial, then

$$h[\overline{x}, y] \simeq \begin{cases} f[\overline{x}] & \Leftarrow \; y = 0 \\ g[\overline{x}, y, h[\overline{x}, y-1]] & \Leftarrow \; \text{o.w.} \end{cases}$$

is partial.

The minimization is partial

If f is partial, then

$g[\overline{x}] \simeq \mu y[f[\overline{x}, y] = 0]$

is partial.

Partial Functions

The primitive recursion is partial

If f and g are partial, then

$$h[\overline{x}, y] \simeq \begin{cases} f[\overline{x}] & \Leftarrow \; y = 0 \\ g[\overline{x}, y, h[\overline{x}, y-1]] & \Leftarrow \; \text{o.w.} \end{cases}$$

is partial.

The minimization is partial

If f is partial, then

$g[\overline{x}] \simeq \mu y[f[\overline{x}, y] = 0]$

is partial.

Partial Functions

Theorem
All the partial functions are computable.

Alternative Definition
Partial functions = Computable functions.

Partial Functions

Theorem
All the partial functions are computable.

Alternative Definition
Partial functions = Computable functions.

Examples

Subtraction is partial

$$f[x,y] \simeq \begin{cases} x - y & \Leftarrow x \geq y \\ \uparrow & \Leftarrow \text{o.w.} \end{cases}$$

$f[x,y] \simeq \mu z[x + y = z]$

Division is partial

$$g[x,y] \simeq \begin{cases} x/y & \Leftarrow \exists k(y.k = x) \\ \uparrow & \Leftarrow \text{o.w.} \end{cases}$$

$g[x,y] \simeq \mu k[k.y = x]$

Examples

Subtraction is partial

$$f[x,y] \simeq \begin{cases} x - y & \Leftarrow \ x \geq y \\ \uparrow & \Leftarrow \ \text{o.w.} \end{cases}$$

$f[x,y] \simeq \mu z[x + y = z]$

Division is partial

$$g[x,y] \simeq \begin{cases} x/y & \Leftarrow \ \exists k(y.k = x) \\ \uparrow & \Leftarrow \ \text{o.w.} \end{cases}$$

$g[x,y] \simeq \mu k[k.y = x]$

Examples

Subtraction is partial

$$f[x,y] \simeq \begin{cases} x - y & \Leftarrow x \geq y \\ \uparrow & \Leftarrow \text{o.w.} \end{cases}$$

$f[x,y] \simeq \mu z[x + y = z]$

Division is partial

$$g[x,y] \simeq \begin{cases} x/y & \Leftarrow \exists k(y.k = x) \\ \uparrow & \Leftarrow \text{o.w.} \end{cases}$$

$g[x,y] \simeq \mu k[k.y = x]$

Examples

Subtraction is partial

$$f[x,y] \simeq \begin{cases} x - y & \Leftarrow \; x \geq y \\ \uparrow & \Leftarrow \; \text{o.w.} \end{cases}$$

$f[x,y] \simeq \mu z[x + y = z]$

Division is partial

$$g[x,y] \simeq \begin{cases} x/y & \Leftarrow \; \exists k(y.k = x) \\ \uparrow & \Leftarrow \; \text{o.w.} \end{cases}$$

$g[x,y] \simeq \mu k[k.y = x]$

Enumeration of the computable functions

Enumeration = Encoding = Effective codding

- Uniqueness: each object has a unique code
- Totality: each natural number is a code of an object
- Effectiveness: For each object one can find algorithmically its code and for each code (number) one can find its object.

Enumeration of the computable functions

Enumeration = Encoding = Effective codding

- Uniqueness: each object has a unique code
- Totality: each natural number is a code of an object
- Effectiveness: For each object one can find algorithmically its code and for each code (number) one can find its object.

Enumeration of the computable functions

Enumeration = Encoding = Effective codding

- Uniqueness: each object has a unique code
- Totality: each natural number is a code of an object
- Effectiveness: For each object one can find algorithmically its code and for each code (number) one can find its object.

Enumeration of the computable functions

Enumeration = Encoding = Effective codding

- Uniqueness: each object has a unique code
- Totality: each natural number is a code of an object
- Effectiveness: For each object one can find algorithmically its code and for each code (number) one can find its object.

Enumeration of the computable functions

Enumeration = Encoding = Effective codding

- Uniqueness: each object has a unique code
- Totality: each natural number is a code of an object
- Effectiveness: For each object one can find algorithmically its code and for each code (number) one can find its object.

Enumeration of the computable functions

- Let $P_0, P_1, \ldots, P_n, \ldots$
  be a list of all the programs (on one variable), and
  $0, 1, \ldots, n, \ldots$ be an effective codding of these programs.
- Each program corresponds to a computable function φ
- Let $\varphi_0, \varphi_1, \ldots, \varphi_n, \ldots$
  be a list of all the computable functions (on one variable), and
  $0, 1, \ldots, n, \ldots$ be an effective codding of these functions.

Enumeration of the computable functions

- Let $P_0, P_1, \ldots, P_n, \ldots$
  be a list of all the programs (on one variable), and
  $0, 1, \ldots, n, \ldots$ be an effective codding of these programs.

- Each program corresponds to a computable function φ

- Let $\varphi_0, \varphi_1, \ldots, \varphi_n, \ldots$
  be a list of all the computable functions (on one variable), and
  $0, 1, \ldots, n, \ldots$ be an effective codding of these functions.

Enumeration of the computable functions

- Let $P_0, P_1, \ldots, P_n, \ldots$
  be a list of all the programs (on one variable), and
  $0, 1, \ldots, n, \ldots$ be an effective codding of these programs.
- Each program corresponds to a computable function φ
- Let $\varphi_0, \varphi_1, \ldots, \varphi_n, \ldots$
  be a list of all the computable functions (on one variable), and
  $0, 1, \ldots, n, \ldots$ be an effective codding of these functions.

Example

Total function which is not computable

$$f[x] \simeq \begin{cases} \varphi_x[x] + 1 & \Leftarrow \varphi_x[x]\downarrow \\ 0 & \Leftarrow \text{o.w.} \end{cases}$$

Assume f is computable. Then $f = \varphi_a$ for some a.
If $a \in Dom[\varphi_a]$ then $\varphi_a[a]\downarrow$. Hence, $f[a] = \varphi_a[a] = \varphi_a[a] + 1$
If $a \notin Dom[\varphi_a]$ then $\varphi_a[a]\uparrow$. Hence, $f[a] = \varphi_a[a] = 0$, but $\varphi_a[a]\uparrow$

Example

Total function which is not computable

$$f[x] \simeq \begin{cases} \varphi_x[x] + 1 & \Leftarrow \ \varphi_x[x] \downarrow \\ 0 & \Leftarrow \ \text{o.w.} \end{cases}$$

Assume f is computable. Then $f = \varphi_a$ for some a.
If $a \in Dom[\varphi_a]$ then $\varphi_a[a] \downarrow$. Hence, $f[a] = \varphi_a[a] = \varphi_a[a] + 1$
If $a \notin Dom[\varphi_a]$ then $\varphi_a[a] \uparrow$. Hence, $f[a] = \varphi_a[a] = 0$, but $\varphi_a[a] \uparrow$

Kleene's S-m-n Theorem

S-m-n Theorem

For any n, m exists a primitive recursive function S^m_n, such that

for any $a, \overline{x}, \overline{y}$

$$\varphi^{(m+n)}_a[\overline{x}, \overline{y}] \simeq \varphi^{(n)}_{S^m_n[a,\overline{x}]}[\overline{y}]$$

Property

Let F be a computable function. Then there exists a number e, such that,

$$F[e, \overline{x}] \simeq \varphi_e[\overline{x}]$$

Property

There exists a number e, such that,

$$e \simeq \varphi_e[\overline{x}]$$

Kleene's S-m-n Theorem

S-m-n Theorem

For any n, m exists a primitive recursive function S^m_n, such that

for any $a, \overline{x}, \overline{y}$

$$\varphi_a^{(m+n)}[\overline{x}, \overline{y}] \simeq \varphi^{(n)}_{S^m_n[a,\overline{x}]}[\overline{y}]$$

Property

Let F be a computable function. Then there exists a number e, such that,

$$F[e, \overline{x}] \simeq \varphi_e[\overline{x}]$$

Property

There exists a number e, such that,

$$e \simeq \varphi_e[\overline{x}]$$

Kleene's S-m-n Theorem

S-m-n Theorem

For any n, m exists a primitive recursive function S^m_n, such that

for any $a, \overline{x}, \overline{y}$

$$\varphi_a^{(m+n)}[\overline{x}, \overline{y}] \simeq \varphi^{(n)}_{S^m_n[a,\overline{x}]}[\overline{y}]$$

Property

Let F be a computable function. Then there exists a number e, such that,

$$F[e, \overline{x}] \simeq \varphi_e[\overline{x}]$$

Property

There exists a number e, such that,

$$e \simeq \varphi_e[\overline{x}]$$

Universal Function

Universal Function Theorem

The universal function

$\Phi_n[a, \overline{x}] \simeq \varphi_a^{(n)}[\overline{x}]$

is computable.

Property

The class of all the total functions on n-variables does not have a computable universal function.

proof

Assume $\Phi[a, \overline{x}]$ is an universal function for the class of all the total functions on one variable.

Let $\varphi[x] \simeq \Phi[x, x] + 1$.

Since Φ is total, φ is also total and hence, there exists a, such that

$\varphi[x] \simeq \Phi[a, x]$.

$\varphi[a] \simeq \Phi[a, a]$, and also $\varphi[a] \simeq \Phi[a, a] + 1$.

Universal Function

Universal Function Theorem

The universal function

$\Phi_n[a, \overline{x}] \simeq \varphi_a^{(n)}[\overline{x}]$

is computable.

Property

The class of all the total functions on n-variables does not have a computable universal function.

proof

Assume $\Phi[a, \overline{x}]$ is an universal function for the class of all the total functions on one variable.

Let $\varphi[x] \simeq \Phi[x, x] + 1$.

Since Φ is total, φ is also total and hence, there exists a, such that

$\varphi[x] \simeq \Phi[a, x]$.

$\varphi[a] \simeq \Phi[a, a]$, and also $\varphi[a] \simeq \Phi[a, a] + 1$.

Universal Function

Universal Function Theorem

The universal function

$\Phi_n[a, \overline{x}] \simeq \varphi_a^{(n)}[\overline{x}]$

is computable.

Property

The class of all the total functions on n-variables does not have a computable universal function.

proof

Assume $\Phi[a, \overline{x}]$ is an universal function for the class of all the total functions on one variable.

Let $\varphi[x] \simeq \Phi[x, x] + 1$.

Since Φ is total, φ is also total and hence, there exists a, such that

$\varphi[x] \simeq \Phi[a, x]$.

$\varphi[a] \simeq \Phi[a, a]$, and also $\varphi[a] \simeq \Phi[a, a] + 1$.

Universal Function

Universal Function Theorem

The universal function

$\Phi_n[a, \overline{x}] \simeq \varphi_a^{(n)}[\overline{x}]$

is computable.

Property

The class of all the total functions on n-variables does not have a computable universal function.

proof

Assume $\Phi[a, \overline{x}]$ is an universal function for the class of all the total functions on one variable.

Let $\varphi[x] \simeq \Phi[x, x] + 1$.

Since Φ is total, φ is also total and hence, there exists a, such that

$\varphi[x] \simeq \Phi[a, x]$.

$\varphi[a] \simeq \Phi[a, a]$, and also $\varphi[a] \simeq \Phi[a, a] + 1$.

Decidable and Semidecidable Sets $A \subseteq \mathbb{N}^n$

Characteristic function of a set χ_A

$$\chi_A[\overline{x}] \simeq \begin{cases} 1 & \Leftarrow \overline{x} \in A \\ 0 & \Leftarrow \text{o.w.} \end{cases}$$

Decidable Set

A set A is decidable iff χ_A is computable.

Decidable and Semidecidable Sets $A \subseteq \mathbb{N}^n$

Characteristic function of a set χ_A

$$\chi_A[\overline{x}] \simeq \begin{cases} 1 & \Leftarrow \overline{x} \in A \\ 0 & \Leftarrow \text{o.w.} \end{cases}$$

Decidable Set

A set A is decidable iff χ_A is computable.

Decidable and Semidecidable Sets $A \subseteq \mathbb{N}^n$

Characteristic function of a set χ_A

$$\chi_A[\overline{x}] \simeq \begin{cases} 1 & \Leftarrow \overline{x} \in A \\ 0 & \Leftarrow \text{o.w.} \end{cases}$$

Decidable Set

A set A is decidable iff χ_A is computable.

Decidable and Semidecidable Sets $A \subseteq \mathbb{N}^n$

Semicharacteristic function of a set C_A

$$C_A[\overline{x}] \simeq \begin{cases} 1 & \Leftarrow \overline{x} \in A \\ \uparrow & \Leftarrow \text{o.w.} \end{cases}$$

Semidecidable Set

A set A is semidecidable iff C_A is computable.

Decidable and Semidecidable Sets $A \subseteq \mathbb{N}^n$

Semicharacteristic function of a set C_A

$$C_A[\overline{x}] \simeq \begin{cases} 1 & \Leftarrow \overline{x} \in A \\ \uparrow & \Leftarrow \text{o.w.} \end{cases}$$

Semidecidable Set

A set A is semidecidable iff C_A is computable.

Decidable and Semidecidable Sets $A \subseteq \mathbb{N}^n$

Theorem
If A is decidable then it is also semidecidable.

Theorem
If A is decidable then $\overline{A}$ is also decidable.

Theorem
If A and B are decidable then
$A \cup B$, $A \cap B$ and $A \backslash B$ are decidable.

Theorem
If A and B are semidecidable then
$A \cup B$ and $A \cap B$ are semidecidable.

Decidable and Semidecidable Sets $A \subseteq \mathbb{N}^n$

Theorem
If A is decidable then it is also semidecidable.

Theorem
If A is decidable then $\overline{A}$ is also decidable.

Theorem
If A and B are decidable then
$A \cup B$, $A \cap B$ and $A \backslash B$ are decidable.

Theorem
If A and B are semidecidable then
$A \cup B$ and $A \cap B$ are semidecidable.

Decidable and Semidecidable Sets $A \subseteq \mathbb{N}^n$

Theorem
If A is decidable then it is also semidecidable.

Theorem
If A is decidable then $\overline{A}$ is also decidable.

Theorem
If A and B are decidable then
$A \cup B$, $A \cap B$ and $A \backslash B$ are decidable.

Theorem
If A and B are semidecidable then
$A \cup B$ and $A \cap B$ are semidecidable.

Decidable and Semidecidable Sets $A \subseteq \mathbb{N}^n$

Theorem
If A is decidable then it is also semidecidable.

Theorem
If A is decidable then $\overline{A}$ is also decidable.

Theorem
If A and B are decidable then
$A \cup B$, $A \cap B$ and $A \backslash B$ are decidable.

Theorem
If A and B are semidecidable then
$A \cup B$ and $A \cap B$ are semidecidable.

Decidable and Semidecidable Sets $A \subseteq \mathbb{N}^n$

Theorem

A set A is semidecidable iff there exists a computable function φ, such that,
$A = Dom[\varphi]$

Post Theorem

A set A is decidable iff A and $\overline{A}$ are semidecidable.

Kleene Set $\mathbb{K}$

The set $\mathbb{K} = \{x \mid \varphi_x[x] \downarrow\}$ is called Kleene set.

Theorem

$\mathbb{K}$ is semidecidable but not decidable.

Decidable and Semidecidable Sets $A \subseteq \mathbb{N}^n$

Theorem

A set A is semidecidable iff there exists a computable function φ, such that,
$A = Dom[\varphi]$

Post Theorem

A set A is decidable iff A and $\overline{A}$ are semidecidable.

Kleene Set $\mathbb{K}$

The set $\mathbb{K} = \{x \mid \varphi_x[x] \downarrow\}$ is called Kleene set.

Theorem

$\mathbb{K}$ is semidecidable but not decidable.

Decidable and Semidecidable Sets $A \subseteq \mathbb{N}^n$

Theorem

A set A is semidecidable iff there exists a computable function φ, such that,
$A = Dom[\varphi]$

Post Theorem

A set A is decidable iff A and $\overline{A}$ are semidecidable.

Kleene Set $\mathbb{K}$

The set $\mathbb{K} = \{x \mid \varphi_x[x] \downarrow\}$ is called Kleene set.

Theorem

$\mathbb{K}$ is semidecidable but not decidable.

Decidable and Semidecidable Sets $A \subseteq \mathbb{N}^n$

Theorem
A set A is semidecidable iff there exists a computable function φ, such that,
$A = Dom[\varphi]$

Post Theorem
A set A is decidable iff A and $\overline{A}$ are semidecidable.

Kleene Set $\mathbb{K}$
The set $\mathbb{K} = \{x \mid \varphi_x[x] \downarrow\}$ is called Kleene set.

Theorem
$\mathbb{K}$ is semidecidable but not decidable.

Outline

Conclusions and Discussion

Halting Problem

There is no program P which may decide for an arbitrary program Q executed on arbitrary input x, whether Q will terminate on x or not.

$$P[Q,x] \simeq \begin{cases} 1 & \Leftarrow \; Q[x]\downarrow \\ 0 & \Leftarrow \; \text{o.w.} \end{cases}$$

$$P[a,x] \simeq \begin{cases} 1 & \Leftarrow \; \varphi_a[x]\downarrow \\ 0 & \Leftarrow \; \text{o.w.} \end{cases}$$

$a \in \mathbb{K} \quad \Leftrightarrow \quad \varphi_a[a]\downarrow \Leftrightarrow \quad P[a,a] = 1$

$a \in \overline{\mathbb{K}} \quad \Leftrightarrow \quad \varphi_a[a]\uparrow \Leftrightarrow \quad P[a,a] = 0$

Thus $\overline{\mathbb{K}}$ is decidable, which is a contradiction.

Conclusions and Discussion

Halting Problem

There is no program P which may decide for an arbitrary program Q executed on arbitrary input x, whether Q will terminate on x or not.

$$P[Q,x] \simeq \begin{cases} 1 & \Leftarrow \; Q[x]\downarrow \\ 0 & \Leftarrow \; \text{o.w.} \end{cases}$$

$$P[a,x] \simeq \begin{cases} 1 & \Leftarrow \; \varphi_a[x]\downarrow \\ 0 & \Leftarrow \; \text{o.w.} \end{cases}$$

$a \in \mathbb{K} \quad \Leftrightarrow \quad \varphi_a[a]\downarrow \Leftrightarrow \quad P[a,a] = 1$

$a \in \overline{\mathbb{K}} \quad \Leftrightarrow \quad \varphi_a[a]\uparrow \Leftrightarrow \quad P[a,a] = 0$

Thus $\overline{\mathbb{K}}$ is decidable, which is a contradiction.

Conclusions and Discussion

Halting Problem

There is no program P which may decide for an arbitrary program Q executed on arbitrary input x, whether Q will terminate on x or not.

$$P[Q, x] \simeq \begin{cases} 1 & \Leftarrow \quad Q[x] \downarrow \\ 0 & \Leftarrow \quad \text{o.w.} \end{cases}$$

$$P[a, x] \simeq \begin{cases} 1 & \Leftarrow \quad \varphi_a[x] \downarrow \\ 0 & \Leftarrow \quad \text{o.w.} \end{cases}$$

$$a \in \mathbb{K} \quad \Leftrightarrow \quad \varphi_a[a] \downarrow \Leftrightarrow \quad P[a, a] = 1$$

$$a \in \overline{\mathbb{K}} \quad \Leftrightarrow \quad \varphi_a[a] \uparrow \Leftrightarrow \quad P[a, a] = 0$$

Thus $\overline{\mathbb{K}}$ is decidable, which is a contradiction.

Conclusions and Discussion

Halting Problem

There is no program P which may decide for an arbitrary program Q executed on arbitrary input x, whether Q will terminate on x or not.

$$P[Q,x] \simeq \begin{cases} 1 & \Leftarrow \quad Q[x]\downarrow \\ 0 & \Leftarrow \quad \text{o.w.} \end{cases}$$

$$P[a,x] \simeq \begin{cases} 1 & \Leftarrow \quad \varphi_a[x]\downarrow \\ 0 & \Leftarrow \quad \text{o.w.} \end{cases}$$

$$a \in \mathbb{K} \quad \Leftrightarrow \quad \varphi_a[a]\downarrow \Leftrightarrow \quad P[a,a] = 1$$

$$a \in \overline{\mathbb{K}} \quad \Leftrightarrow \quad \varphi_a[a]\uparrow \Leftrightarrow \quad P[a,a] = 0$$

Thus $\overline{\mathbb{K}}$ is decidable, which is a contradiction.

Conclusions and Discussion

Halting Problem

There is no program P which may decide for an arbitrary program Q executed on arbitrary input x, whether Q will terminate on x or not.

$$P[Q,x] \simeq \begin{cases} 1 & \Leftarrow \quad Q[x]\downarrow \\ 0 & \Leftarrow \quad \text{o.w.} \end{cases}$$

$$P[a,x] \simeq \begin{cases} 1 & \Leftarrow \quad \varphi_a[x]\downarrow \\ 0 & \Leftarrow \quad \text{o.w.} \end{cases}$$

$$a \in \mathbb{K} \quad \Leftrightarrow \quad \varphi_a[a]\downarrow \Leftrightarrow \quad P[a,a] = 1$$

$$a \in \overline{\mathbb{K}} \quad \Leftrightarrow \quad \varphi_a[a]\uparrow \Leftrightarrow \quad P[a,a] = 0$$

Thus $\overline{\mathbb{K}}$ is decidable, which is a contradiction.