MATHEMATICS FOR INFORMATION SCIENCE

# NO.5 TURING MACHINE AND COMPUTABILITY

Tatsuya Hagino

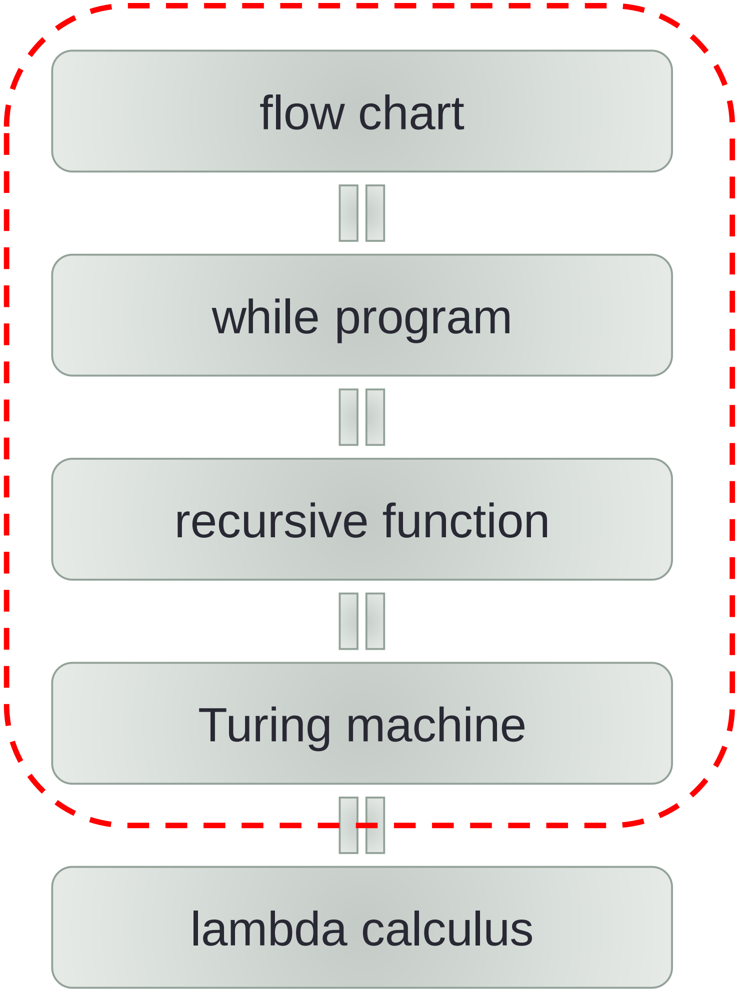
hagino@sfc.keio.ac.jp

Slides URL

https://vu5.sfc.keio.ac.jp/slide/

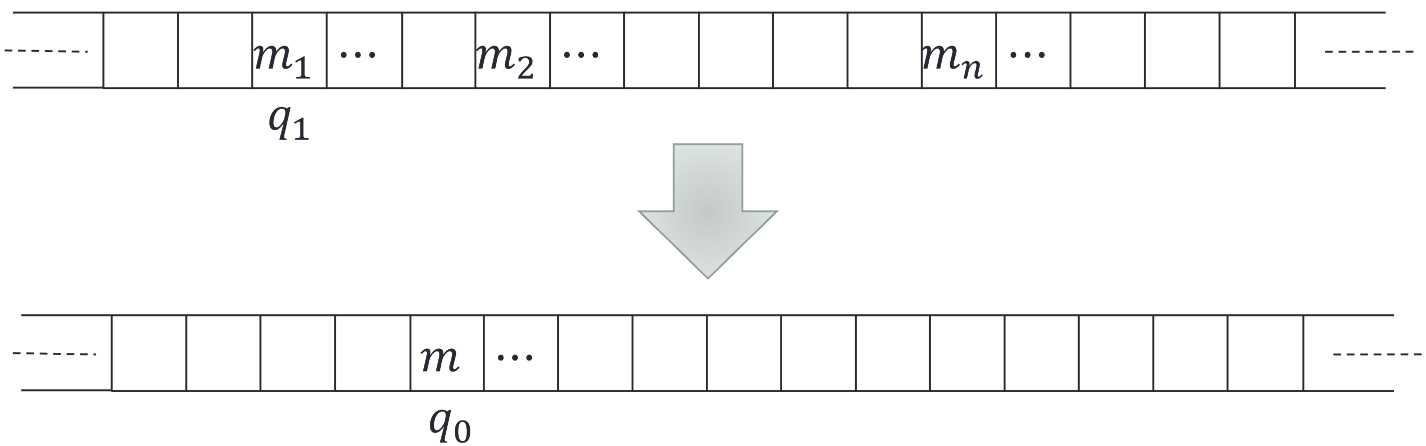
# So far

- Computation
  - flow chart program
  - while program
  - recursive function
    - primitive recursive function
    - minimization operator
  - Turing machine



# Computation

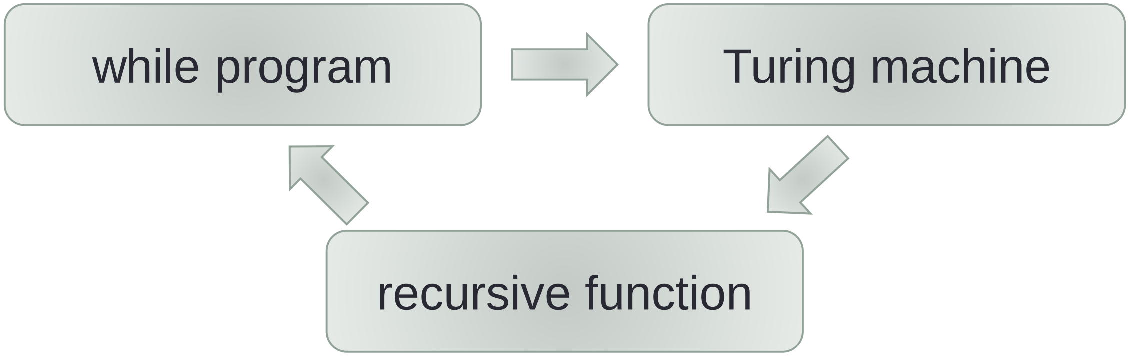
- A Turing machine $M$ computes $f: N^n \rightarrow N$ when:
  - Place $m_1, m_2, \cdots, m_n$ on the tape with decimal numbers separated with a blank
  - Start $M$ with the head at the leftmost number position.
  - When $M$ terminates, the number at the head is the decimal number of $f(m_1, m_2, \cdots, m_n)$.



# Computation and Program

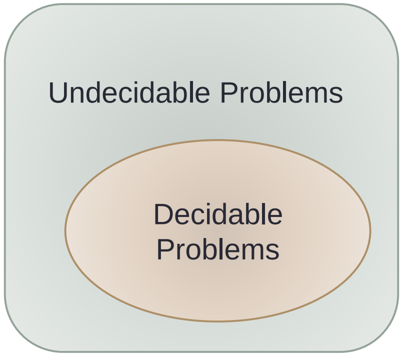
- A Turing machine may not terminate.
  - The function it computes is not total, but partial.

- **Theorem**
  - If a Turing machine can compute $f: N^n \to N$, it can be computed by a while program.
  - If $f: N^n \to N$ is a recursive function, there is a Turing machine which can compute the same function.



# Decidable vs Undecidable Problems

- Decidable Problem
  - A problem for which a program can say yes or no.
  - The program needs to terminate.
  - The corresponding recursive function needs to be total.




- Undecidable Problem
  - A problem which is not decidable.
  - There might be a program which may say yes, but it does not termination if the answer is no.
  - The corresponding function is not recursive, or it is recursive but not total.

- **Halting Problem:**
  - Is there a program which tells whether a given program $P$ for a given input $a_1, \ldots, a_n$ will eventually terminate and return a value or will run forever?

# Encoding Programs

- In order to make a program as an input to another program, we need to represent a program as a number (i.e. encoding)
- Encoding flow chart programs:
  - Boxes are connected by arrows
  - Put a number to each box
  - Each box is one of the following:
    - $\mathrm{input}(x_1, x_2, \cdots, x_n)$
    - $x_i := m$
    - $x_i := x_j + x_k$
    - $x_i := x_j - x_k$
    - $x_i := x_j \times x_k$
    - $x_i := x_j \div x_k$
    - $\mathrm{if}\ (x_i = x_j)$
    - $\mathrm{output}(x_i)$

# Encoding

- Let $x_1, \dots, x_n$ be input variables and $x_{n+1}, x_{n+2}, \dots, x_t$ be other variables.
- Let $A_1, A_2, \dots, A_l$ be boxes of program $P$ where $A_1$ is the input box and $A_l$ is the output box.
- Using Gödel number, encode each box as $\#A$:

| $A_a$ | $\#A_a$ |
|---|---|
| $\text{input}(x_1, x_2, \cdots, x_n)$ | $\langle 1, n, a' \rangle$ |
| $x_i := m$ | $\langle 2, i, m, a' \rangle$ |
| $x_i := x_j + x_k$ | $\langle 3, i, j, k, a' \rangle$ |
| $x_i := x_j - x_k$ | $\langle 4, i, j, k, a' \rangle$ |
| $x_i := x_j \times x_k$ | $\langle 5, i, j, k, a' \rangle$ |
| $x_i := x_j \div x_k$ | $\langle 6, i, j, k, a' \rangle$ |
| $\text{if}\left(x_i = x_j\right)$ | $\langle 7, i, j, a', a'' \rangle$ |
| $\text{output}(x_i)$ | $\langle 8, i \rangle$ |

- The program can be encoded as:
  - $\#P = \langle \#A_1, \#A_2, \dots, \#A_l \rangle$

# Interpreter for $P$

**Theorem**:

- The following partial function $\mathrm{comp}_n \colon N^{n+1} \to N$ is computable.

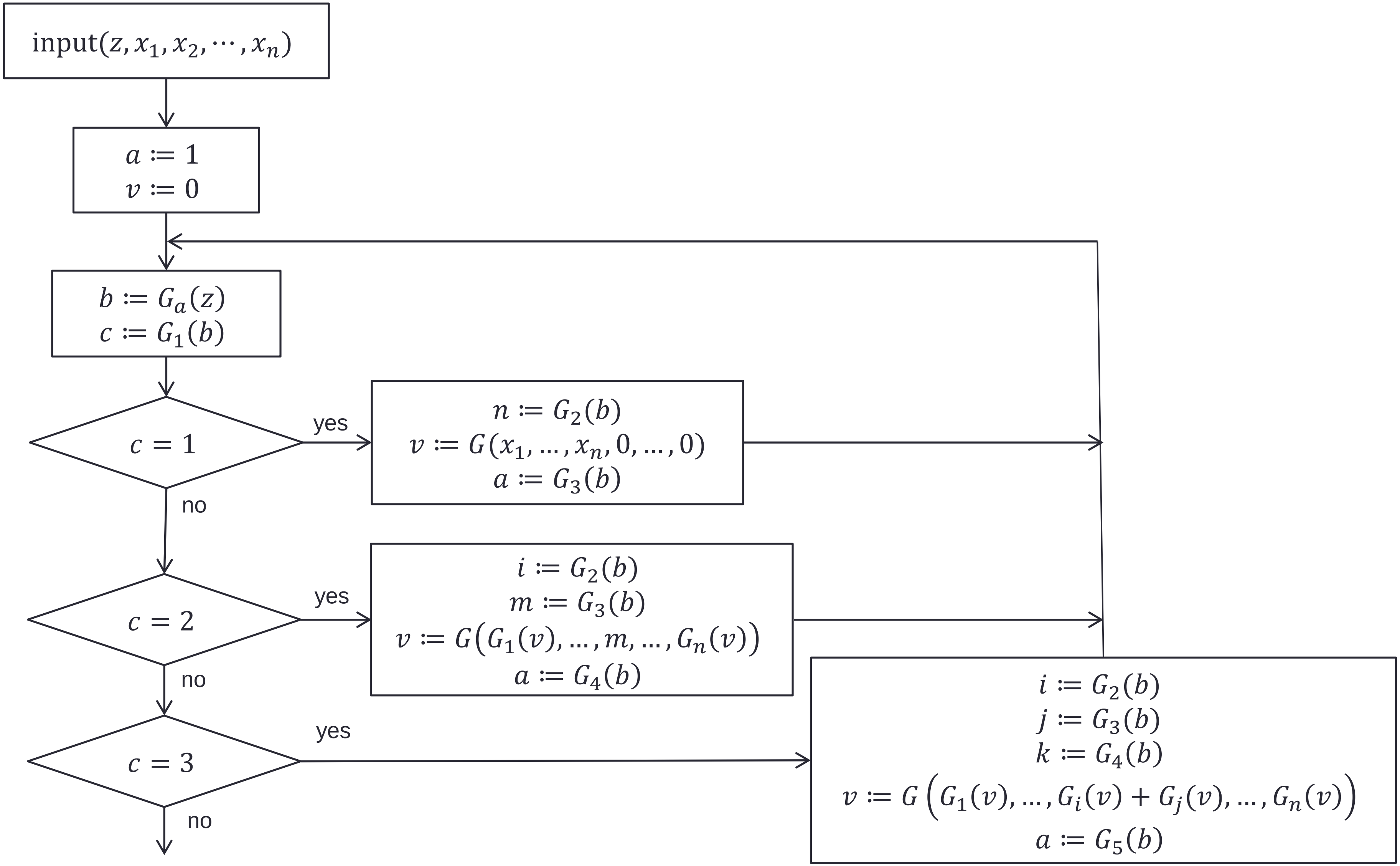
$$\mathrm{comp}_n(z, x_1, \dots, x_n) = \begin{cases} y & \text{when } z = \#P \text{ and } y = f_P(x_1, \dots x_n) \\ \text{undefined} & \text{otherwise} \end{cases}$$

where $f_P$ is the recursive function for program $P$.

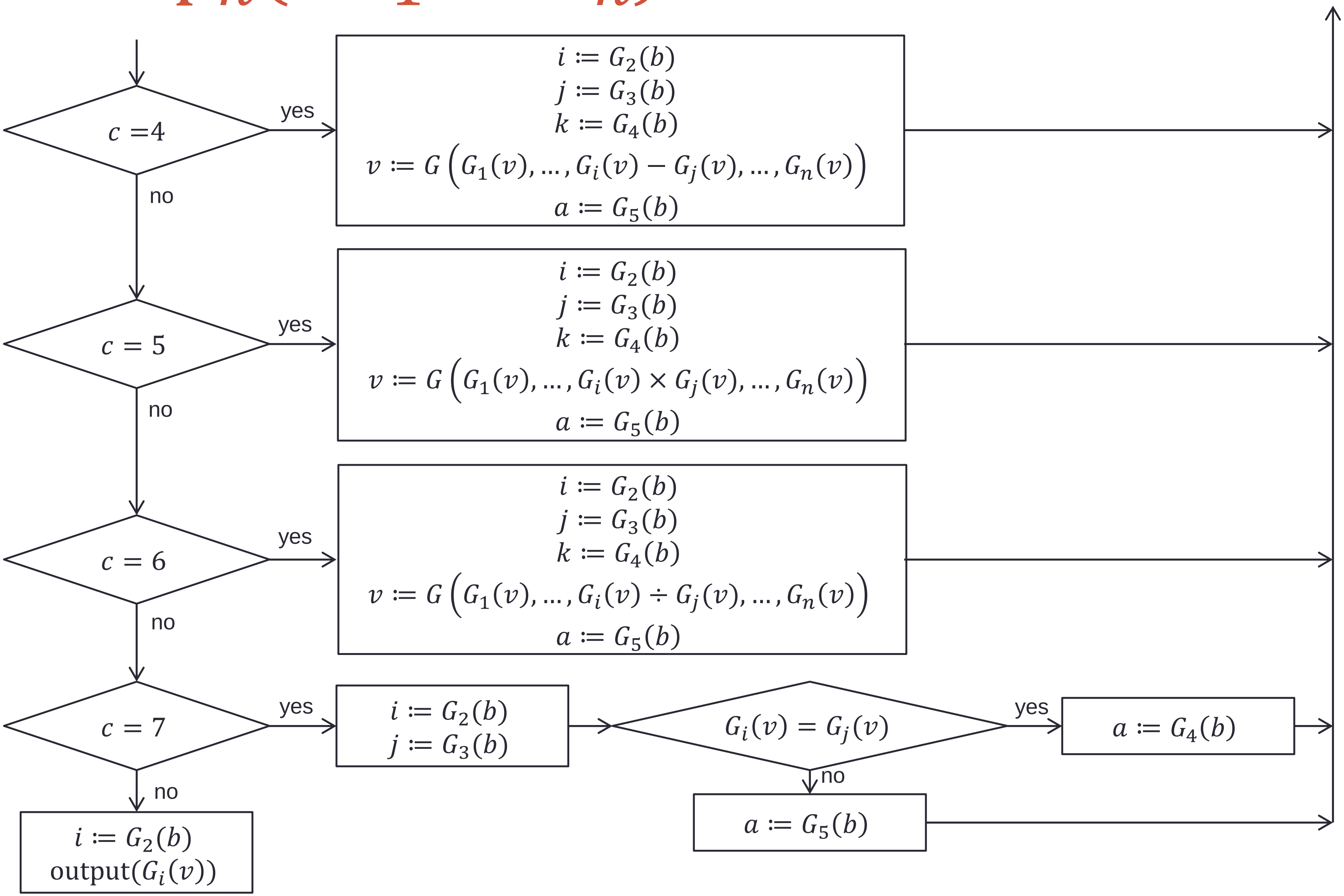
**Proof**:

- Write a program which computes $\mathrm{comp}_n$ by simulating the flow chart program represented by $\#P$.

# $\mathrm{comp}_n(z, x_1, \dots, x_n)$



# $\mathrm{comp}_n(z, x_1, \ldots, x_n)$ cont.



# Is comp Total?

**Theorem:** If $\mathrm{comp}_n: N^{n+1} \to N$ is extended to a total function

$$g: N^{n+1} \to N$$

$g$ is not recursive.

**Proof:**

- Show the case for $n = 1$:
- Proof by contradiction and use Cantor's diagonal argument.
- Assume $\mathrm{comp}_1(z, x) = g(z, x)$ and $g: N^2 \to N$ is a total recursive function.
- Let $h(x) = g(x, x) + 1$. Then, $h$ is also a total recursive function.
- There is a program which calculates $h$.
- Let $c$ be the code.
- Then, from the definition of $\mathrm{comp}_1$, $h(x) = \mathrm{comp}_1(c, x)$.
- Give $h$ an input $c$.

$$h(c) = \mathrm{comp}_1(c, c) = g(c, c)$$

- This contradicts with $h(c) = g(c, c) + 1$.
- Therefore, a recursive total function $g$ does not exist. (QED)

# Recursive Predicate

**Definition:** Predicate $p: N^n \to \{T, F\}$ is a recursive predicate if its characteristic function $C_p: N^n \to N$ is recursive.

- $C_p$ is total.
- $p$ is decidable.

- If $p(x_1, \dots, x_n), q(x_1, \dots, x_n)$ and $r(x_1, \dots, x_n, y)$ are recursive, the following predicates are also recursive:
  - $p(x_1, \dots, x_n) \wedge q(x_1, \dots, x_n)$
  - $p(x_1, \dots, x_n) \vee q(x_1, \dots, x_n)$
  - $\neg p(x_1, \dots, x_n)$
  - $\forall z < y\big(r(x_1, \dots, x_n, z)\big)$
  - $\exists z < y\big(r(x_1, \dots, x_n, z)\big)$

# Halting Problem is Undecidable

- Define predicate $\mathrm{halt}_\mathrm{n}(z, x_1, \ldots, x_n) : N^{n+1} \to \{T, F\}$ as follows:

$$\mathrm{halt}_n(z, x_1, \ldots, x_n) = \begin{cases} T & \text{when } \mathrm{comp}_n(z, x_1, \ldots, x_n) \text{ is defined} \\ F & \text{when } \mathrm{comp}_n(z, x_1, \ldots, x_n) \text{ is undefined} \end{cases}$$

**Theorem:** $\mathrm{halt}_\mathrm{n}(z, x_1, \ldots, x_n)$ is not recursive (i.e. undecidable).

**Proof:**

- If $\mathrm{halt}_n(z, x_1, \ldots, x_n)$ is a recursive predicate, its characteristic function $C_{\mathrm{halt}_n}$ is recursive and total. Then,

$$g(z, x_1, \ldots, x_n) = C_{\mathrm{halt}_n}(z, x_1, \ldots, x_n) \times \mathrm{comp}_n(z, x_1, \ldots, x_n)$$

is a total recursive function and this contradicts with the previous theorem. (QED)

# Totality Problem is Undecidable

**Theorem:** For $n > 0$, there is no total recursive function $g: N^{n+1} \rightarrow N$ which satisfies the following:

$$\{ g(c, x_1, \dots, x_n): N^{n+1} \rightarrow N \mid c \in N \} = \{ f: N^n \rightarrow N \mid f \text{ is total and recursive} \}$$

- $\mathrm{comp}_n(z, x_1, \dots, x_n): N^{n+1} \rightarrow N$ is the universal function for recursive functions (both partial and total), but there is no universal function for total recursive functions.

**Proof:**
- In the case for $n = 1$, if $g: N^2 \rightarrow N$ exists, $f(x) = g(x, x) + 1$ is a total recursive function.
- Let $c$ be the code of $f$, $g(c, x) = f(x) = g(x, x) + 1$ and this contradicts when $x = c$.
- In the case for $n > 1$, the proof can be similar. (QED).

**Corollary:** $\mathrm{total}_n(z) \equiv \forall x_1 \cdots \forall x_n \left(\mathrm{halt}_n(z, x_1, \dots, x_n)\right)$ is not a recursive predicate, i.e. $\mathrm{total}_n(z)$ is undecidable.

**Proof:** If $C_{\mathrm{total}_n}$ is the characteristic function of $\mathrm{total}_n$,

$$g(z, x_1, \dots, x_n) = C_{\mathrm{total}_n}(z) \times \mathrm{comp}_n(z, x_1, \dots, x_n)$$

$g$ is a total recursive function and this contradicts with previous theorem. (QED)

# Undecidable Predicates

- $\mathrm{halt}_n(z, x_1, \dots, x_n)$
  - whether a give program $z$ terminates for the input $x_1, \dots, x_n$ or not.
- $\mathrm{total}_n(z)$
  - whether a given program $z$ always terminates or not.
- $\forall x_1 \cdots \forall x_n (\mathrm{comp}_n(z, x_1, \dots, x_n) = 0)$
  - whether a given program $z$ always outputs 0 or not.
- $\exists x_1 \cdots \exists x_n (\mathrm{comp}_n(z, x_1, \dots, x_n) = 0)$
  - whether a given program $z$ outputs 0 for some input or not.
- For $z$, the domain of $\mathrm{comp}_n(z, x_1, \dots, x_n)$ is finite.
  - whether a program $z$ terminates for finite sets of input or not.
- For $z$, $\mathrm{comp}_n(z, x_1, \dots, x_n)$ is a constant function.
  - whether a program $z$ outputs always the same number or not.
- For $z$ and $z'$, $\mathrm{comp}_n(z, x_1, \dots, x_n) = \mathrm{comp}_n(z', x_1, \dots, x_n)$
  - whether two programs $z$ and $z'$ are same or not.

# s-m-n Theorem

**Theorem:** For natural numbers $m$ and $n$, there is a primitive recursive function $S_{m,n}: N^{m+1} \to N$ which satisfies:

$$\mathrm{comp}_{m+n}(z, x_1, \dots, x_n, y_1, \dots, y_m) = \mathrm{comp}_n\big(S_{m,n}(z, y_1, \dots, y_m), x_1, \dots, x_n\ \big)$$

**Proof:** $S_{m,n}(z, u_1, \dots, u_m)$ is the function which converts

$$z = \langle \#A_1, \#A_2, \dots, \#A_l \rangle$$

into

$$z' = \big\langle \#\big(\mathrm{input}(x_1, \dots, x_n)\big), \#(y_1 := u_1), \dots, \#(y_m := u_m), \#A_2, \dots, \#A_l \big\rangle$$

which represents:

- $\mathrm{input}(x_1, \dots, x_n)$
- $y_1 := u_1$
- $\dots$
- $y_m := u_m$
- $A_2$
- $\dots$
- $A_l$

The conversion function can be written as a primitive recursive function. (QED)

# Recursion Theorem

**Theorem:** For $n$ and a total recursive function $f: N \to N$, there is a natural number $c$ which makes the following equation true:

$$\mathrm{comp}_n(f(c), x_1, \dots, x_n) = \mathrm{comp}_n(c, x_1, \dots, x_n)$$

**Proof:**

- Let $a$ be the code for $\mathrm{comp}_{n+1}(y, x_1, \dots, x_n, y)$.
- $\mathrm{comp}_{n+1}(y, x_1, \dots, x_n, y) = \mathrm{comp}_{n+1}(a, x_1, \dots, x_n, y) = \mathrm{comp}_n\left(S_{1,n}(a, y), x_1, \dots, x_n\right)$
- Let $b$ be the code for $\mathrm{comp}_n\left(f\left(S_{1,n}(a, y)\right), x_1, \dots, x_n\right)$
- $\mathrm{comp}_n\left(f\left(S_{1,n}(a, y)\right), x_1, \dots, x_n\right) = \mathrm{comp}_{n+1}(b, x_1, \dots, x_n, y)$
- $\mathrm{comp}_n\left(f\left(S_{1,n}(a, b)\right), x_1, \dots, x_n\right) = \mathrm{comp}_{n+1}(b, x_1, \dots, x_n, b) = \mathrm{comp}_n\left(S_{1,n}(a, b), x_1, \dots, x_n\right)$
- $c = S_{1,n}(a, b)$

(QED)

# Rice Theorem

**Theorem:** Let $n$ be a natural number. If a predicate $p(z)$ satisfies the following two conditions, $p(z)$ is not recursive (i.e. $p(z)$ is undecidable).

*(1)* $\forall c \forall c' \left( \forall x_1 \cdots \forall x_n \left( \mathrm{comp}_n(c, x_1, \dots, x_n) = \mathrm{comp}_n(c', x_1, \dots, x_n) \right) \Rightarrow p(c) \equiv p(c') \right)$

*(2)* $\exists c \exists c' \left( p(c) \land \neg p(c') \right)$

- (1) means that $p(z)$ truth value is the same for the same program.
- (2) means that $p(z)$ is true for certain number and is false for a different number.

**Proof:**

- If $p$ is a recursive predicate, let $C_p$ be its characteristic function.
- Let define $f\colon N \to N$ using $c$ and $c'$ which satisfy (2) as follows:

$$f(z) = C_p(z) \times c' + \left(1 - C_p(z)\right) \times c$$

- From the definition, $p\left(f(z)\right) \not\equiv p(z)$
- Since $f$ is a total recursive function, using recursion theory there exists $c''$ which makes $\mathrm{comp}_n(f(c''), x_1, \dots, x_n) = \mathrm{comp}_n(c'', x_1, \dots, x_n)$.
- From (1), $p(f(c'')) = p(c'')$, but this contradicts. (QED)

- Using this theorem, we can prove many predicates are undecidable.
  - $p(z) \equiv$ "$\mathrm{comp}_n(z, x_1, \dots, x_n)$ is a constant function."
  - $p(z)$ is same for the same program, and there are a constant program and a not-constant one.

# Post Correspondence Problem

**Problem:** Given a finite set of string pairs,

$$\{(s_1, t_1), (s_2, t_2), \dots, (s_n, t_n)\}$$

using string concatenation, determine whether there is a number sequence $i_1, \dots, i_m$ which makes the following equality hold:

$$s_{i_1} s_{i_2} \dots s_{i_m} = t_{i_1} t_{i_2} \dots t_{i_m}$$

**Example:**

- $\{(e, abcde), (ababc, ab), (d, cab)\}$

$$\boxed{a\,b\,a\,b\,c}\boxed{a\,b\,a\,b\,c}\boxed{d}\boxed{e}$$
$$\boxed{a\,b}\boxed{a\,b}\boxed{c\,a\,b}\boxed{a\,b\,c\,d\,e}$$

- This problem (post correspondence problem) is undecidable.
  - There is no program which gives a solution to the problem or none if there is no solution.

# Homework 5

- Solve the following post correspondence problem.
  - $\{(abb, a), (b, abb), (a, bb)\}$

  - If there is an answer, please show how you combine them to create the same string.
  - If there is no answer, please explain why.

# Summary

- Decidable Problem
  - A problem for which a program can say yes or no.

- Undecidable Problem
  - A problem which is not decidable.

- Undecidable predicates:
  - Halting problem
  - Totality problem
  - Post correspondence problem