Computation and the Halting Problem

Nohman Akhtari

May 2, 2024

Table of Contents

Unlimited Register Machines

Unlimited Register Machines

Given a finite sequence of natural numbers, we have

$$\begin{array}{ccc} r_1 & r_2 & r_3 \\ \boxed{3} & \boxed{6} & \boxed{10} \; \ldots \end{array}$$

Given a finite sequence of natural numbers, we have

$$\begin{array}{ccc} r_1 & r_2 & r_3 \\ \boxed{3} & \boxed{6} & \boxed{10} \end{array} \dots$$

$$\boxed{4}\,\boxed{6}\,\boxed{10} \dots \qquad (I_1)$$

Unlimited Register Machines

Given a finite sequence of natural numbers, we have

$$\begin{array}{ccc} r_1 & r_2 & r_3 \\ \boxed{3} & \boxed{6} & \boxed{10} \ \ldots \end{array}$$

$$\boxed{4}\,\boxed{6}\,\boxed{10}\ \ldots \qquad (I_1)$$

$$\boxed{4}\,\boxed{0}\,\boxed{10}\ \ldots \qquad (I_2)$$

Unlimited Register Machines

Given a finite sequence of natural numbers, we have

r_1	r_2	r_3	
3	6	10	...

4	6	10	...	(I_1)

$\boxed{4}$	0	10	...	(I_2)

Register Operations

For registers $i, j, q \in \mathbb{N}^+$, we define

- $Z(i)$: Set r_i to 0
- $S(i)$: Increment r_i by 1
- $T(i, j)$: Transfer r_i to r_j
- $J(i, j, q)$: If $r_i = r_j$, execute q

Unlimited Register Machines

Unlimited Register Machine (URM)

A URM M is a finite sequence of instructions $M = (I_1, I_2 \ldots)$.

Unlimited Register Machines

Unlimited Register Machine (URM)

A URM M is a finite sequence of instructions $M = (I_1, I_2 \ldots)$.

Any URM M can represented through a recursively defined a (total) function

$$\varphi_M : \mathbb{N}^n \to \mathbb{N}.$$

Unlimited Register Machines

Gödel Coding

Any URM $M = (I_1, I_2, \ldots)$ can be assigned a unique natural number through a method called Gödel Coding that uses prime factorization:

- We use the number $\langle 0, i \rangle = 2 \cdot 3^{i+1}$ for $Z(i)$
- We use the number $\langle 1, i \rangle = 2^2 \cdot 5^{i+1}$ for $S(i)$
- We use the number $\langle 2, i, j \rangle = 2^3 \cdot 3^{i+1} \cdot 5^{j+1}$ for $T(i, i)$
- We use the number $\langle 3, i, j, q \rangle = 2^4 \cdot 3^{i+1} \cdot 5^{j+1} \cdot 7^{q+1}$ for $J(i, j, q)$

We obtain the Gödel Code of M, which is simply a number, by calculating

$$G\ddot{o}delCode(M) = \langle Number(I_1), Number(I_2), \ldots \rangle.$$

For example, $G\ddot{o}delCode(J(1, 1, 1)) = \langle \langle 3, 1, 1, 1 \rangle \rangle$.

Unlimited Register Machines

Gödel Coding

Any URM $M = (I_1, I_2, \ldots)$ can be assigned a unique natural number through a method called Gödel Coding that uses prime factorization:

- We use the number $\langle 0, i \rangle = 2 \cdot 3^{i+1}$ for $Z(i)$
- We use the number $\langle 1, i \rangle = 2^2 \cdot 5^{i+1}$ for $S(i)$
- We use the number $\langle 2, i, j \rangle = 2^3 \cdot 3^{i+1} \cdot 5^{j+1}$ for $T(i, i)$
- We use the number $\langle 3, i, j, q \rangle = 2^4 \cdot 3^{i+1} \cdot 5^{j+1} \cdot 7^{q+1}$ for $J(i, j, q)$

We obtain the Gödel Code of M, which is simply a number, by calculating

$$GödelCode(M) = \langle Number(I_1), Number(I_2), \ldots \rangle.$$

For example, $GödelCode(J(1, 1, 1)) = \langle \langle 3, 1, 1, 1 \rangle \rangle$.

$$\varphi_M = \varphi_e : \mathbb{N}^n \to \mathbb{N}$$

Unlimited Register Machines

1. What is a Universal Machine?

1. What is a Universal Machine?
2. Does such a machine exist?

Unlimited Register Machines

1. What is a Universal Machine?
2. Does such a machine exist?
3. Why should we care?

Unlimited Register Machines

Universal Machine

A Universal Machine is a Unlimited Register Machine M_U that can compute the output for any $n \in \mathbb{N}$ and any machine M.

Unlimited Register Machines

Universal Machine

A Universal Machine is a Unlimited Register Machine M_U that can compute the output for any $n \in \mathbb{N}$ and any machine M.

If such a machine exists, there exists a sequence of instructions M_U such that we can depict any arbitrary machine M with it!

Unlimited Register Machines

Universal Machine

A Universal Machine is a Unlimited Register Machine M_U that can compute the output for any $n \in \mathbb{N}$ and any machine M.

If such a machine exists, there exists a sequence of instructions M_U such that we can depict any arbitrary machine M with it!

The surprise is that such a machine actually exists: The Gödel Code e for the universal machine is such that given some Gödel Code e' of any machine, it can simulate that machine e'.

Unlimited Register Machines

Universal Machine

A Universal Machine is a Unlimited Register Machine M_U that can compute the output for any $n \in \mathbb{N}$ and any machine M.

If such a machine exists, there exists a sequence of instructions M_U such that we can depict any arbitrary machine M with it!

The surprise is that such a machine actually exists: The Gödel Code e for the universal machine is such that given some Gödel Code e' of any machine, it can simulate that machine e'.

In simple terms, there exists a machine that can calculate any other machine! We let φ be the universal machine, we will use the notation

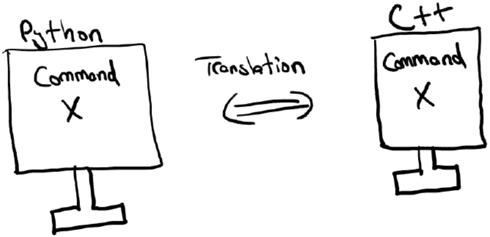
$$\varphi_{(e,\vec{x})}.$$

Unlimited Register Machines

One consequence of φ_e is that any Turing-complete computer can understand another computer because the languages can be translated!

Unlimited Register Machines

One consequence of φ_e is that any Turing-complete computer can understand another computer because the languages can be translated!



The Halting Problem

The Halting Problem

What is the output of the machine $J(1, 1, 1)$ on input $(1, 1) = \vec{x} \in \mathbb{N}^2$?

The Halting Problem

What is the output of the machine $J(1,1,1)$ on input $(1,1) = \vec{x} \in \mathbb{N}^2$?

Since the computation would loop forever (= not halt), we denote the outputs as $\uparrow$. If not, the computation has a number as output (= halt).

The Halting Problem

What is the output of the machine $J(1,1,1)$ on input $(1,1) = \vec{x} \in \mathbb{N}^2$?

Since the computation would loop forever ($=$ not halt), we denote the outputs as $\uparrow$. If not, the computation has a number as output ($=$ halt).

Halting Problem

Given a machine M with code e and an input $\vec{x} \in \mathbb{N}^n$, the Halting Problem is the problem of determining whether the computation $\varphi_{(e,\vec{x})}$ will halt or not halt.

The Halting Problem

What is the output of the machine $J(1, 1, 1)$ on input $(1, 1) = \vec{x} \in \mathbb{N}^2$?

Since the computation would loop forever (= not halt), we denote the outputs as $\uparrow$. If not, the computation has a number as output (= halt).

Halting Problem

Given a machine M with code e and an input $\vec{x} \in \mathbb{N}^n$, the Halting Problem is the problem of determining whether the computation $\varphi_{(e,\vec{x})}$ will halt or not halt.

Next, we will now combine the idea of halting idea with the universal machine $\varphi_{(e,\vec{x})}$.

The Halting Problem

Suppose we can compute any halting problem for any machine and any input. We can then define the following set:

Halting Set

The set

$$K = \{e \in \mathbb{N} : \varphi_e(e) \text{ halts}\}$$

is called the Halting Set.

The Halting Problem

Suppose we can compute any halting problem for any machine and any input. We can then define the following set:

Halting Set

The set

$$K = \{e \in \mathbb{N} : \varphi_e(e) \text{ halts}\}$$

is called the Halting Set.

The Halting Problem

$\varphi_e(x)$	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7
2	$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$
3	1	1	1	1	1	1	1	1
4	2	0	$\uparrow$	0	0	0	0	0
5	1	$\uparrow$	3	$\uparrow$	5	$\uparrow$	7	$\uparrow$
6	0	1	1	2	3	5	8	13
7	8	0	7	15	$\uparrow$	0	3	$\uparrow$

The Halting Problem

$\varphi_e(x)$	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7
2	$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$
3	1	1	1	1	1	1	1	1
4	2	0	$\uparrow$	0	0	0	0	0
5	1	$\uparrow$	3	$\uparrow$	5	$\uparrow$	7	$\uparrow$
6	0	1	1	2	3	5	8	13
7	8	0	7	15	$\uparrow$	0	3	$\uparrow$

Assume the halting set is computable, we can compute all diagonal entries that halt, and set the non-halting computations $(= \uparrow)$ to 0.

The Halting Problem

This means that we can define a function

$$f(e) = \begin{cases} \varphi_e(e) + 1 & \text{, if } e \in K \\ 0 & \text{, otherwise} \end{cases}$$

that is computable, but not in our table by Cantor's Diagonal Argument. Hence, no row contains the above computable function, which is a contradiction. $\perp$

The Halting Problem

This means that we can define a function

$$f(e) = \begin{cases} \varphi_e(e) + 1 & \text{, if } e \in K \\ 0 & \text{, otherwise} \end{cases}$$

that is computable, but not in our table by Cantor's Diagonal Argument. Hence, no row contains the above computable function, which is a contradiction. $\perp$

$\boxed{\text{The Halting Set is not computable and Halting Problem is unsolvable!}}$

Topics around the Halting Problem

1. Gödel's Incompleteness Theorem

Topics around the Halting Problem

1. Gödel's Incompleteness Theorem
2. Every algorithm can be translated into URM (Church-Turing Thesis)

Topics around the Halting Problem

1. Gödel's Incompleteness Theorem
2. Every algorithm can be translated into URM (Church-Turing Thesis)
3. Can't avoid bugs in computers since we would need to avoid non-halting computations!

Topics around the Halting Problem

1. Gödel's Incompleteness Theorem
2. Every algorithm can be translated into URM (Church-Turing Thesis)
3. Can't avoid bugs in computers since we would need to avoid non-halting computations!
4. Is human thought a computation?

Big thank you to ...

- Jin, for being a great and sapient mentor,
- Avik, for motivating me to keep doing math,
- Maxine and Leo, for organizing the DRP,
- and Gödel, for causing a mess in math!