

SYMMETRY AS TURING MACHINE - APPROACH TO SOLVE L VS P VS NP

KOBAYASHI, KOJI

1. ABSTRACT

This article describes about that L is not P and P is not NP by using difference of symmetry each problems.

Deterministic Turing Machine (DTM) change configuration by using transition functions. This changing keep halting configuration, therefore these configurations set are classification by DTM as equivalence relation. The view of classification, there are different between L and P. L can compute equivalence class that cardinals is polynomial size, but P can compute exponential size. Therefore, L cannot compute P-Complete problems, and L is not P. And using L is not P, we can prove P is not NP. All P problem have equivalent reversible function and DTM can reduce from NP-Complete problem to another NP-Complete problem by using this reversible function. If P is NP, equivalent Logarithm space reduction exists. But that means L is P and contradict L is not P. Therefore, P is not NP.

2. PREPARATION

In this article, we use description as follows;

Definition 1. We will use the term “ L ” as L problem set, “ P ” as P problem set, “ $P - Complete$ ” as P-Complete problem set, “ $NP - Complete$ ” as NP-Complete problem set, “ FL ” as Logarithm space function set, “ FP ” as Polynomial time function set.

“ DTM ” as Deterministic Turing Machine set. “ $LDTM$ ” as Turing Machine set that compute L and FL , “ $pDTM$ ” as Turing Machine set that compute P and FP . “ $RpDTM$ ” as Reversible $pDTM$.

And we will use words and theorems of References [1, 2, 3] in this paper.

3. SYMMETRY AS TURING MACHINE

Show the symmetry as DTM. Transition functions of DTM are deterministic, therefore DTM compute only one next configuration. Because this transition keep halting configuration, these configuration make equivalence class that equivalence relation is DTM. But this equivalence class is limited to the tape size. LDTM information without input tape (working tape, head position, state) is atmost $O(\log n)$. Therefore, LDTM can compute atmost $O(n^c)$ cardinals equivalence class.

Theorem 2. *LDTM can compute atmost $O(n^c)$ cardinals equivalence class. That is, LDTM can read input that cardinals is atmost $O(n^c)$ and write output that cardinals is atmost $O(n^c)$.*

Proof. Number of state that LDTM can be capable is atmost $O(n^c)$. Therefore, LDTM can pick out atmost $O(n^c)$ states and cannot pick out more than $O(n^c)$ states. Therefore, this theorem was shown. \square

4. L IS NOT P

Prove $L \neq P$ by using LDTM limitation. Mentioned above 2, LDTM can compute atmost $O(n^c)$ cardinals equivalence class. But P-Complete problem have equivalence class that is more than $O(n^c)$ cardinals. Therefore LDTM cannot compute P-Complete problem.

Definition 3. We will use the term “*CIRCUIT – VALUE*” as CIRCUIT-VALUE problem set. To facilitate, all partial circuit in $p \in \text{CIRCUIT – VALUE}$ (without input values) already simplified. Therefore, if circuit input value is not given, circuit is minimum syntax of p .

Theorem 4. $L \neq P$

Proof. We prove it using reduction to absurdity. We assume that $L = P$. Therefore, $m \in \text{LDTM}$ can compute $p \in \text{CIRCUIT – VALUE}$.

Think about circuit size that m can compute at a time. Mentioned above 2, m can compute atmost $O(n^c)$ cardinals equivalence class. m compute equivalence class of p syntax, therefore circuit size that m can compute at a time is atmost $O(\log n)$. And mentioned above3, if m can simplify p without input value, it is contradict *CIRCUIT – VALUE* condition. Therefore, m must use input value to simplify p .

Think about that m compute circuit that size become more than $O(\log n)$. m can compute atmost $O(\log n)$ size circuit, therefore m must reduce and replace circuit to compute more than $O(\log n)$ size circuit. And m cannot decide circuit output that input is not decided, therefore m must compute the circuit that connect p input and write these output to working tape. But p has $O(n)$ width and depth, and m must keep output in working tape that space amount to $O(n)$. Therefore m cannot reach compute p output value from p input value and contradict $L = P$.

Therefore, this theorem was shown than reduction to absurdity. \square

5. P IS NOT NP

Prove $P \neq NP$ by using $L \neq P$.

Theorem 5. $P \neq NP$

Proof. We prove it using reduction to absurdity. We assume that $P = NP$. Therefore, all $p, q \in NP – Complete$ have $f \in \text{LDTM}$ that reduce p to q .

$$\forall p, q \in NP – Complete \exists f \in \text{LDTM} (f(p) = q)$$

If $p \in NP – Complete, g \in \text{RpDTM}$, then

$$p \leq_p g(p)$$

and

$$g(p) \leq_p g^{-1}(g(p)) = p \in NP$$

Therefore,

$$g(p) \in NP – Complete$$

That is,

$$\forall p \in NP – Complete \forall g \in \text{RpDTM} \exists f \in \text{LDTM} (f(p) = g(p))$$

But mentioned above4, $\text{RpDTM} \neq \text{LDTM}$ and contradict it.

Therefore, this theorem was shown than reduction to absurdity. \square

REFERENCES

- [1] Michael Sipser, (translation) OHTA Kazuo, TANAKA Keisuke, ABE Masayuki, UEDA Hiroki, FUJIOKA Atsushi, WATANABE Osamu, Introduction to the Theory of COMPUTATION Second Edition, 2008
- [2] OGIHARA Mitsunori, Hierarchies in Complexity Theory, 2006
- [3] MORITA Kenichi, Reversible Computing, 2012