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. That is, DTM classfy these configuration into equivalence class. The view of equivalence class, 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 reductcion 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 problem set, "FP" as Polynomial time function problem 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 infomation 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. □

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 make easy, all partial circuit in $p \in CIRCUIT-VALUE$ (without input values) already simplified. Therefore, if circuit input values are not given, circuit is minimum syntax of p.

CIRCUIT-VALUE syntax have many cardinals equivalence class. TM necessary to decide some gate input to decide gate output. Therefore CIRCUIT-VALUE syntax have minimum circuit to decide CIRCUIT-VALUE output. Minimum circuit become representative of cardinals equivalence class, and size of minimum circuit type amount to $O(c^n)$. Therefore CIRCUIT-VALUE syntax have $O(c^n)$ size cardinals equivalence class.

Definition 4. We will use the term "Minimum circuit" as $p \in CIRCUIT-VALUE$ that output does not change if any $\vee$ gate input add some gate output and any $\wedge$ gate input delete. Therefore, any $\vee$ gate have only one input and any $\wedge$ gate have all input.

Theorem 5. *DTM must classfy minimum circuit syntax to compute* $p \in CIRCUIT-VALUE$.

Proof. We prove it using reduction to absurdity. We assume that DTM can compute $p \in CIRCUIT-VALUE$ without classfying minimum circuit syntax. Therefore DTM compute cannot classfy some minimum circuit syntax.

Let C is minimum circuit syntax set that DTM cannot classfy. If $p, q \in C$ output are different each other, then DTM cannot classfy p, q output. Therefore, all $p, q \in C$ output necessary to output same value. But $\neg p$ also have same minimum circuit syntax except output NOT gate. Therefore $\neg p \in C$. That is, DTM cannot classfy $p, \neg p$ and contradict that TM can compute $p \in CIRCUIT-VALUE$.

Therefore, this theorem was shown than reduction to absurdity. □

Theorem 6. *Cardinals equivalence class of minimum circuit syntax amount to* $O(c^n)$ *size.*

Proof. Any minimum circuit syntax can add NOT gate each input. These minimum circuit structure become another minimum circuit structure each other. Therefore minimum circuit amount to $O(c^n)$ size. □

Theorem 7. $L \neq P$

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

Think about circuit size that m can compute. Mentioned above 2, m can compute atmost $O(n^c)$ cardinals equivalence class. But mentioned above 56, m must classfy

$O(c^n)$ cardinals equivalence class to compute $p \in CIRCUIT-VALUE$. Therefore m cannot compute p and contradict $L = P$.

Therefore, this theorem was shown than reduction to absurdity. □

5. P is not NP

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

Theorem 8. $P \neq NP$

Proof. We prove it using reduction to absurdity. We assume that $P = NP$, therefore all $p, q \in NP - Complete$ have $f \in LDTM$ that reduce p to q.

$\forall p, q \in NP - Complete \exists f \in LDTM\,(f(p) = q)$

If $p \in NP - Complete$ and $g \in RpDTM$ then

$p \leq_p g(p)$

and

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

Therefore

$g(p) \in NP - Complete$

That is,

$\forall p \in NP - Complete \forall g \in RpDTM \exists f \in LDTM\,(f(p) = g(p))$

But mentioned above7, $RpDTM \neq LDTM$ and contradict it.

Therefore, this theorem was shown than reduction to absurdity. □

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