SYMMETRY AS TURING MACHINE - APPROACH TO SOLVE P VS NP

KOJI KOBAYASHI

1. Abstract

This article describes about that P is not NP by using difference of symmetry.

Turing Machine (TM) change configuration by using transition functions. This changing keep halting configuration. That is, TM classify these configuration into equivalence class. The view of equivalence class, there are different between P and coNP. Some coNP problem have over polynomial size totally order inputs. These problem cannot reduce P because these totally order must keep. Therefore we cannot reduce some coNP problem to P problem. This means P is not NP.

2. Computation forest

In this article, we will use words and theorems of References [1, 2, 3] in this paper. About problem, we use description as follows;

Definition 1. We will use the term "Input" as data that Turing Machine compute, "Output" as result that Turing Machine compute. "Problem" as set of all input that same Turing Machine can compute same output.

To simplify TM ability, define computation list and computation forest.

Definition 2. We will use the term "Computation list" as lists of transition function target value that appear computation history.

Theorem 3. *Deterministic TM(DTM) configure directed forest graph of conputation list. Nondeterministic TM(NTM) configure Directed acyclic graph of computation list. The graph root is halting configuration and the leaves are start configuration. Each vertex have $O(1)$ degree.*

Proof. This is trivial. All conputation list have anchor as halting configuration. TM can generate this graph that have root as halting configuration by using transition functions. Number of transition functions is $O(1)$. Therefore degree of this graph is $O(1)$. □

Definition 4. We will use the term "Computation forest" as graph of computation list of whole TM that roots are halting configurations. We will write tree structure as nesting of leafs. That is, node that have A, B leaf is (A, B). And we will write edge as $\rightarrow$. That is, edge between A and (A, B) is $A \rightarrow (A, B)$.

That is, TM classify input by using transition functions. But TM have limitation that transition functions can classify only $O(1)$, TM necessary to take time to classify some problems.

Theorem 5. *Some TM have conputation forest that minimum graph is* $(((A,B),C),D)$. *That is, some TM have totally order of inputs.*

Proof. To think computation list $a = A \to (((A,B),C),D)$. This computation list have 3 partical computation list

$a_1 = A \to (A,B)$

$a_2 = (A,B) \to ((A,B),C)$

$a_3 = ((A,B),C) \to (((A,B),C),D)$

Each a_1, a_2, a_3 classify each configuration. If TM cannot use some of a_1, a_2, a_3, TM cannot classify A and another input and TM is not equal. Therefore, TM must use a_1, a_2, a_3 to classify A.

It is same that TM compute $b = B \to (((A,B),C),D)$. This computation list have 3 partical computation list

$b_1 = B \to (A,B)$

$a_2 = (A,B) \to ((A,B),C)$

$a_3 = ((A,B),C) \to (((A,B),C),D)$

and b_1, a_2, a_3 is necesarry to compute B.

It is same $c = C \to (((A,B),C),D)$ and $d = D \to (((A,B),C),D)$. These partical computation list are

$c_2 = C \to ((A,B),C)$

$a_3 = ((A,B),C) \to (((A,B),C),D)$

and

$d_3 = D \to (((A,B),C),D)$

Each $a_1, a_2, a_3, b_1, c_2, d_3$ cannot delete from computation forest. It is necessary to classify each A, B, C, D to other inputs. Therefore computation forest is $(((A,B),C),D)$ and this theorem was shown. □

3. P is not NP

Prove $P \neq NP$ by using totally order of coNP inputs. Mentioned above 5, some input have totally order defined by computation forest. This totally order must keep when coNP problem reduce to P problem. But coNP problem have totally order that become over $O(n^c)$ size. Therefore we cannot reduce coNP problem to P. This means $P \neq NP$.

Theorem 6. *Some coNP problem have totally order inputs set that become over* $O(n^c)$ *size.*

Proof. To think coNP computation forest that compute $\overline{3SAT}$ problem. NTM compute forking phase that determine truth value set and verifying phase that verify the truth value set make input formula true. This computation list depend on formula's Minimal Unsatisfiable Core(MUC). If MUC A,B,C,D description is

$A = a_0 \wedge a_1 \wedge a_2 \wedge a_3$

$B = b_0 \wedge a_1 \wedge a_2 \wedge a_3$

$C = c_0 \wedge c_1 \wedge a_2 \wedge a_3$

$D = d_0 \wedge d_1 \wedge d_2 \wedge a_3$

then computation forest have totally order that correspond to $(((A,B),C),D)$. And we can make another MUC by permuting some clauses of the MUC. These number of permuted fomula become over $O(n^c)$ size. Therefore this theorem was shown. □

Theorem 7. $P \neq NP$

Proof. We prove it using reduction to absurdity. We assume that $P = NP$, therefore we can reduce all coNP problem to P problem in polynomial time.

But mentioned above 6, coNP have totally input order that become over $O(n^c)$ size. And mentioned above 5, minimum computation forest must include this totally input order. That is, computation forest have input that computation list become over $O(n^c)$ size and we cannot reduce coNP to P in polynomial time. 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
[4] TANAKA Kazuyuki, SUZUKI Toshio, Mathematical Logic and Set, 2003, p.58