INPUT RELATION AND COMPUTATIONAL COMPLEXITY

KOJI KOBAYASHI

Abstract. This paper describes about complexity of PH problems by using "Almost all monotone circuit family" and "Accept input pair that sandwich reject inputs".

Explained in Michael Sipser "Introduction to the Theory of COMPUTATION", circuit family that emulate Deterministic Turing machine (DTM) are almost all monotone circuit family except some NOT-gate that connect input variables (like negation normal form (NNF)). Therefore, we can find out DTM limitation by using this "NNF Circuit family".

To clarify NNF circuit family limitation, we pay attention to AND-gate and OR-gate relation. If two accept "Neighbor input" pair that sandwich reject "Boundary input" in Hamming distance, NNF circuit have to meet these different values of neighbor inputs in AND-gate to differentiate boundary inputs. NNF circuit have to use unique AND-gate to identify such neighbor input.

The other hand, we can make neighbor input problem "Neighbor Tautology DNF problem (NTD)" in PH. NTD is subset of tautology DNF that do not become tautology if proper subset of one variable permutate positive / negative. NTD include neighbor input pair which number is over polynomial size of input length. Therefore NNF circuit family that compute NTD are over polynomial size of length, and NTD that include PH is not in P.

1. NNF circuit family

Explained in [Sipser] Circuit Complexity section, Circuit family can emulate DTM only using NOT-gate in changing input values $\{0,1\}$ to $\{01,10\}$. This "almost all monotone circuit family" have simple structure like monotone circuit family.

Definition 1.1.

We use term as following;

Date: 2018-03-14.

NNF : Negation Normal Form.

DTM : Deterministic Turing Machine

DNF : Disjunctive Normal Form.

In this paper, we will use words and theorems of References [Sipser].

To simplify, we treat DNF as set of clauses, and also treat clause as set of literals as far as it's all right with you.

Definition 1.2.

We will use the term;

"NNF Circuit Family" as circuit family that have no NOT-gate except connecting input gates directly (like negation normal form). DTM emulator which mentioned Book [Sipser] Circuit Complexity section are included in NNF Circuit family. To simplify, circuit can compute shorter input from circuit input (such shorter input have filler with concrete input).

"Input variable pair" as output pair of input gate and NOT-gate $\{01, 10\}$ that correspond to an input variable $\{0, 1\}$.

"Accept input" as input that circuit family output 1.

"Reject input" as input that circuit family output 0.

"Neighbor input" as accept inputs that no accept inputs exists between these accept input with Hamming distance.

"Boundary input of neighbor input" as reject inputs that exist between neighbor inputs with Hamming distance.

"Different Variables" as subset of input variables that difference each other in neighbor input.

"Same Variables" as subset of input variables that same each other in neighbor input.

"Effective circuit of input t" as one of minimal sub circuit of NNF circuit that decide circuit output as 1 with input t. Effective circuit do not include gate even if gate change output 0 and effective circuit keep output 1. To simplify, effective circuit do not include NOT-gate (monotone circuit).

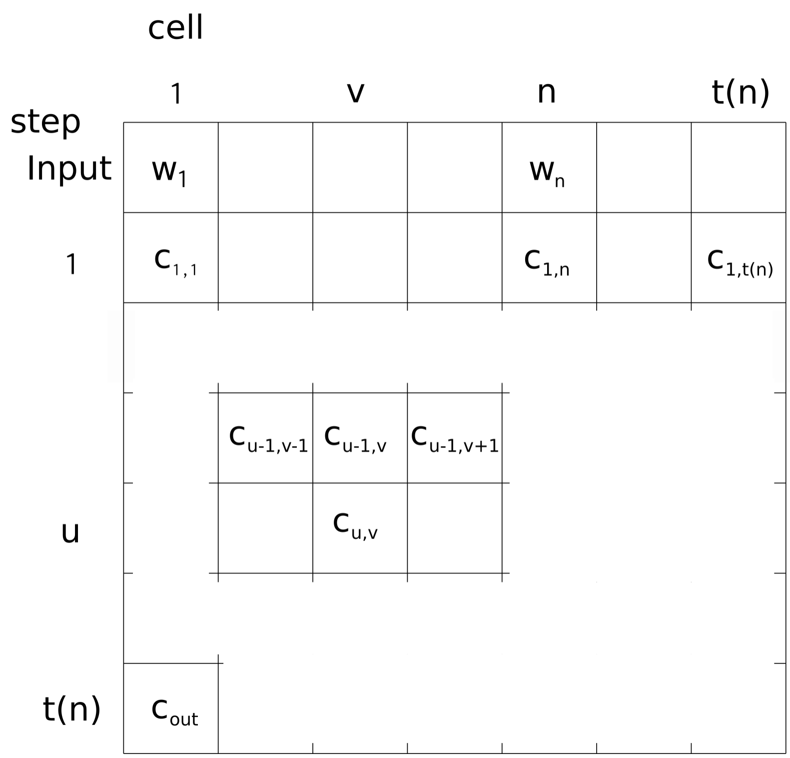

FIGURE 1.1. NNF circuit block diagram

Theorem 1.3.

Let $t : N \longrightarrow N$ be a function where $t(n) \geq n$.

If $A \in TIME(t(n))$ then NNF circuit family can emulate DTM that compute A with $O\left(t^2(n)\right)$ gate.

Proof. This Proof is based on [Sipser] proof.

NNF circuit family can emulate DTM by computing every step's cell values (and head state if head on the cell). Figure 1.1 shows part of a NNF circuit block diagram.

Input of this circuit is modified $w_1 \cdots w_n$ to $c_{1,1} \cdots c_{1,n}$, and finally output result at $c_{out} = c_{t(n),1}$ cell. This circuit emulate DTM behavior, so $c_{u,v}$ compute cell's state of step u from previous step cell $c_{u-1,v}$ and each side cells $c_{u-1,v-1}, c_{u-1,v+1}$ (because head affect atmost side cells in each step).

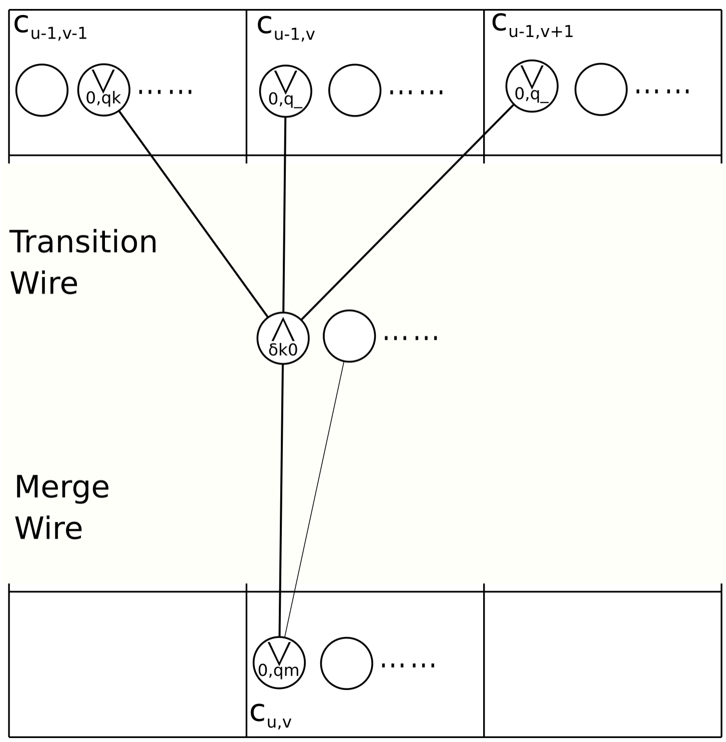

FIGURE 1.2. $c_{u,v}$ circuit

Figure 1.2 shows part of $c_{u,v}$ circuit that implement transition function δ_{k0} "if state is q_k and tape value is 0, then move +1 and change state to q_m". This circuit shows one of transition configuration which $(c_{u-1,v}, c_{u-1,v+1}) = (0, 0)$. q_{-} means "no head on the cell".

Each OR-gate $\vee_{w,q}$ in $c_{u,v}$ correspond to every step's cell condition (cell value w, and head status q if head exist on the $c_{u,v}$ cell), and output 1 if and only if $c_{u,v}$ cell satisfy corresponding condition. Previous step's $\vee$ output in $c_{u-1,v-1}$, $c_{u-1,v}$, $c_{u-1,v+1}$ are connected to next step's AND-gate $\wedge_\delta$ in $c_{u,v}$ with transition wire. Each $\wedge_\delta$ correspond to transition function δ, and each $\wedge_\delta$ output correspond to each transition function's result of $c_{u,v}$. So $\wedge_\delta$ in $c_{u,v}$ output 1 if and only if previous step's $\vee$ output in $c_{u-1,v-1}$, $c_{u-1,v}$, $c_{u-1,v+1}$ satisfy transition function δ condition. Each transition functions affect (or do not affect) next step's condition,

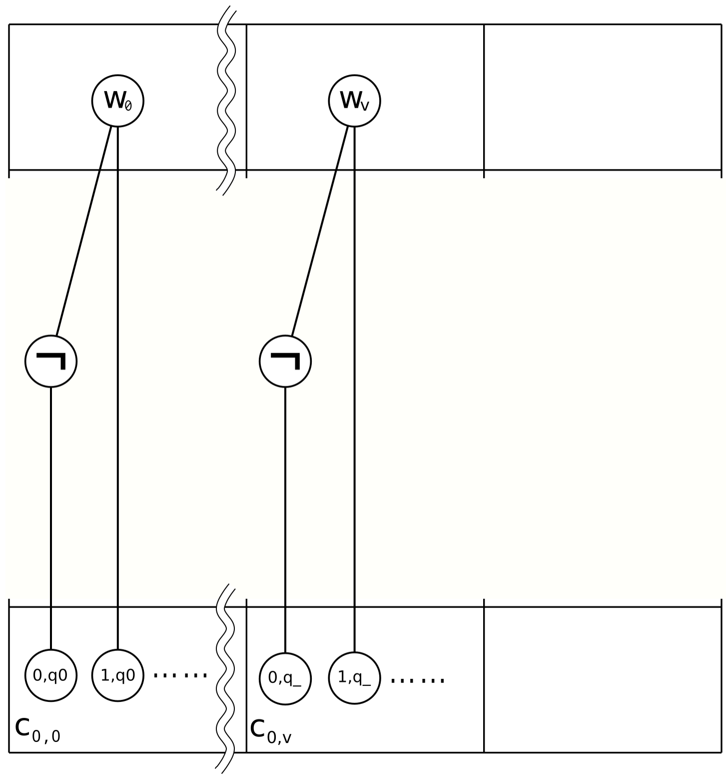

FIGURE 1.3. First step

so $\wedge_\delta$ output is connected to $\vee_{w,qm}$ in $c_{u,v}$ and decide $c_{u,v}$ condition. Because DTM have constant number of transition functions, NNF can compute each step's cell by using constant number of AND-gates and OR-gates (without NOT-gate).

First step's cells are handled in a special way. Input is $\{0,1\}^*$ and above monotone circuit cannot manage 0 value. So NNF circuit compute $\{0,1\}^* \longrightarrow \{01,10\}^*$ by using NOT-gate.

□

Corollary 1.4.

NNF circuit family can compute P problem with polynomial number of gates of input length.

Confirm NNF circuit family behavior. NNF circuit family can emulate DTM with polynomial number of gate of DTM computation time. All effective circuit become DAG that root is specified OUTPUT-gate. All gates that include effective circuit become 1 if OUTPUT-gate become 1. Especially, all different variables of input cannot overlay in same input, so all effective circuits (with different inputs) are join at OR-gate to connect OUTPUT-gate. This NNF circuit behavior clarify input exclusivity and symmetry.

Theorem 1.5.

All input variable pair of different variables join OR-gate in effective circuit.

Proof. Because input variable pair does not become 1 in same input, so it is necessary to join OR-gate and output 1 to connect OUTPUT-gate in effective circuit. □

Theorem 1.6.

NNF circuit have at least one unique AND-gate to differentiate neighbor input and boundary input.

Proof. Mentioned above 1.5, all accept input variable pair of different variables join at OR-gate. Because NNF circuit is almost all monotone circuit, there is two case of joining at OR-gate;

a) all different variables meet at AND-gate, and join at OR-gate after meeting AND-gate.

b) some partial different variables meet at AND-gate, and join at OR-gate these AND-gate output, and meet at AND-gate all OR-gate output.

Case a), some AND-gate become 1 if and only if input include one side of different variables. Therefore, trunk of these AND-gate does not become 1 if AND-gate does not include these different variables as input of sub circuit that connect trunk.

Case b), because no boundary input become accept input, some OR-gate which join neighbor input become 0 if input include boundary input. That is, effective circuit become 0 if some of these OR-gate become 0, and become 1 if all of these

OR-gate become 1. Therefore, it is necessary that effective circuit include AND-gate that meet all these OR-gate which join different variables (and other same variables). Such AND-gate correspond to each different variables pair. So AND-gate is differ from each different variables pair. □

That is to say, neighbor input with sandwitch structure cannot permutate proper partial input of different variables . NNF circuit have to use unique AND-gate to differentiate neighbor input and boundary inputs.

2. Neighbor Tautology DNF

Let clarify number of neighbor input pair. To consider DNF tautology problem, some input become neighbor input by changing one variable positive / negative. So we define new partial problem of DNF tautology.

Definition 2.1.

We will use the term "Neighbor Tautology DNF problem" or "NTD" as partial Minimal Tautology DNF problem which input also tautology if one variables x change positive / negative $\{x, \overline{x}\} \rightarrow \{\overline{x}, x\}$, and not tautology if proper subset of one variables x change positive / negative.

$$NTD = \left\{ f \mid f \equiv \top, f \begin{pmatrix} \cdots & x & \overline{x} & \cdots \\ \cdots & \overline{x} & x & \cdots \end{pmatrix} \equiv \top, g = f \begin{pmatrix} \cdots & \{x,x\} & \{\overline{x},\overline{x}\} & \cdots \\ \cdots & \{x,\overline{x}\} & \{\overline{x},x\} & \cdots \end{pmatrix} \not\equiv \top \right\}$$

$\begin{pmatrix} \cdots & x & \overline{x} & \cdots \\ \cdots & \overline{x} & x & \cdots \end{pmatrix}$: permutate all $x, \overline{x}$ to $\overline{x}, x$.

$\begin{pmatrix} \cdots & \{x,x\} & \{\overline{x},\overline{x}\} & \cdots \\ \cdots & \{x,\overline{x}\} & \{\overline{x},x\} & \cdots \end{pmatrix}$: permutate (any) proper subset of $x, \overline{x}$ to $\overline{x}, x$.

Theorem 2.2.

If $f \in NTD$, then $f \begin{pmatrix} \cdots & x & \overline{x} & \cdots \\ \cdots & \overline{x} & x & \cdots \end{pmatrix}$ is neighbor input of f.

Proof. It is trivial because of $x, \overline{x}$ symmetry with tautology and NTD definition;

$$f \begin{pmatrix} \cdots & x & \overline{x} & \cdots \\ \cdots & \overline{x} & x & \cdots \end{pmatrix} \begin{pmatrix} \cdots & x & \overline{x} & \cdots \\ \cdots & \overline{x} & x & \cdots \end{pmatrix} = f \equiv \top$$

$$f\begin{pmatrix} \cdots & x & \overline{x} & \cdots \\ \cdots & \overline{x} & x & \cdots \end{pmatrix}\begin{pmatrix} \cdots & \{x,x\} & \{\overline{x},\overline{x}\} & \cdots \\ \cdots & \{x,\overline{x}\} & \{\overline{x},x\} & \cdots \end{pmatrix}$$

$$= f\begin{pmatrix} \cdots & \{x,x\} & \{\overline{x},\overline{x}\} & \cdots \\ \cdots & \{x,\overline{x}\} & \{\overline{x},x\} & \cdots \end{pmatrix} \not\equiv \top \qquad \square$$

Theorem 2.3.

Minimal Tautology DNF (MTD) correspond to NTD.

Proof. Proof this theorem by constructing NTD from MTD.

If $f \in MTD$ and $f \notin NTD$, then there are some variable x that keep tautology to change proper subset of x.

$$f \in MTD \wedge f \notin NTD \rightarrow \exists x\left(f\begin{pmatrix} \cdots & x & \overline{x} & \cdots \\ \cdots & \{x,\overline{x}\} & \{x,\overline{x}\} & \cdots \end{pmatrix} \equiv \top\right)$$

Let attach free variable y to $\overline{x}$. y have some relation g with x.

$$f \in MTD \wedge f \notin NTD \rightarrow \exists x\left(\left(f\begin{pmatrix} \cdots & x & \overline{x} & \cdots \\ \cdots & \{x,\overline{y}\} & \{x,\overline{y}\} & \cdots \end{pmatrix} \equiv \top\right) \wedge (g(x,y) \equiv \top)\right)$$

y: free variable.

However, from $f \in MTD$ then

$(x,y) \rightarrow (1,1),(0,0)$

and from $f\begin{pmatrix} \cdots & x & \overline{x} & \cdots \\ \cdots & \{x,\overline{y}\} & \{x,\overline{y}\} & \cdots \end{pmatrix} \equiv \top$ then

$(x,y) \rightarrow (1,0),(0,1)$

So

$(x,y) \rightarrow (1,1),(0,0),(1,0),(0,1)$

and g is no bind of (x,y). So

$$f \in MTD \wedge f \notin NTD \rightarrow \exists x\left(f\begin{pmatrix} \cdots & x & \overline{x} & \cdots \\ \cdots & \{x,\overline{y}\} & \{x,\overline{y}\} & \cdots \end{pmatrix} \equiv \top\right)$$

This means

$$f \in MTD \wedge f \notin NTD \rightarrow \exists x\left(f\begin{pmatrix} \cdots & x & \overline{x} & \cdots \\ \cdots & \{x,\overline{y}\} & \{x,\overline{y}\} & \cdots \end{pmatrix} \in MTD\right)$$

y: free variable.

On the other hand, each MTD have limitation of length and number of variables type. So we can repeat this operation to any proper subset of variables cannot change another free variable. Such MTD satisfy NTD condition. □

x, y of NTD that made by 2.3 is independent each other, but we can modify easily to depend x, y each other. Before proofing this, we proof following lemma.

Lemma 2.4.

There is some DNF f which;

a) become 1 at one of any set of truth value assignment T

$\forall T \forall t \in T\left(f\left(t\right)=1\right)$

b) each clauses have pre-defined 3 variables combination. (We can only decide these literal become positive or negative.)

c) number of clauses is atmost polynomial size of variables type.

Proof. Let $f=d_1 \vee d_2 \vee \cdots \vee d_n$ that variables is $x_1, x_2, \cdots x_k$ and $n=O\left(k^c\right)$, and d_1 include variables x_1, x_2, x_3. Because we can decide positive / negative of x_1, x_2, x_3 in d_1, so d_1 is possible 8 patterns;

$x_1 \wedge x_2 \wedge x_3$, $\overline{x_1} \wedge x_2 \wedge x_3$, $x_1 \wedge \overline{x_2} \wedge x_3$, $\overline{x_1} \wedge \overline{x_2} \wedge x_3$,

$x_1 \wedge x_2 \wedge \overline{x_3}$, $\overline{x_1} \wedge x_2 \wedge \overline{x_3}$, $x_1 \wedge \overline{x_2} \wedge \overline{x_3}$, $\overline{x_1} \wedge \overline{x_2} \wedge \overline{x_3}$

These possible d_1 become partition of truth value assignment, one of above d_1 become true at least $\frac{1}{8}$ of truth value assignment T. So we can reduce number of $|T|$ atmost $\frac{7}{8}$ by deciding suitable positive / negative pattern as d_1.

Above condition is applicable another clauses $d_2, \cdots d_n$, so we can decide positive / negative of variables $x_1, x_2, \cdots x_k$ in $d_2, \cdots d_n$ one by one to reduce T atmost $\frac{7}{8}$. Number of $|T|$ is at most 2^k, so some constant c_0 that $2^k \times (7/8)^{n^{c_0}} \rightarrow 0$. Therefore, we can make f that cover $\forall T \forall t \in T\left(f\left(t\right)=1\right)$. □

Theorem 2.5.

Any NTD can convert some NTD that have all pair of variables in some clauses, and number of these clauses is atmost polynomial of variables types.

Proof. If NTD f does not have clauses which include both x and y, we can make another NTD f' that include x, y in same clause with following step;

1) add literal y or $\overline{y}$ to some clauses c that include $x, \overline{x}$

$c \to c' = c \wedge Y \mid Y \in \{y, \overline{y}\}$

$c = X \wedge \cdots \mid X \in \{x, \overline{x}\}$

2) add new clauses d which include x, y and complement all truth value assignment $\{t\}$ that $c'(t) = 0 \to d(t) = 1$.

Mentioned above 2.4, number of such clauses is atmost polynomial number of variables type. So $|f'|$ is polynomial size of $|f|$ because number of variables type in f is atmost $|f|$. $\square$

Theorem 2.6.

There is some NTD f that does not keep same clauses to permutate literal $x, \overline{x}$.

$$\exists f \in NTD \left(f \cap f \begin{pmatrix} \cdots & x & \overline{x} & \cdots \\ \cdots & \overline{x} & x & \cdots \end{pmatrix} \neq f \cup f \begin{pmatrix} \cdots & x & \overline{x} & \cdots \\ \cdots & \overline{x} & x & \cdots \end{pmatrix} \right)$$

$f \begin{pmatrix} \cdots & x & \overline{x} & \cdots \\ \cdots & \overline{x} & x & \cdots \end{pmatrix}$: *clauses that permutate all $x, \overline{x}$ to $\overline{x}, x$ in f.*

Proof. To modify methods mentioned above proof 2.5, we can easily make f from which keep same clauses to permutate literal $x, \overline{x}$. In 2) step, we choose some variables set that do not same variables set in any clauses (and also another clauses in f), these clauses does not become symmetry with permutation of $(x, \overline{x})$. $\square$

Theorem 2.7.

$NTD \in PH$

Proof. We can solve NTD by computing;

a) input as TAUT problem, and

b) all input that change any proper subset of one type literal as non TAUT problem.

We can compute b) that choice changing literal as universal and compute them as non TAUT problem. coNP Oracle machine with TAUT oracle can compute this problem. Therefore NTD is in PH. □

Theorem 2.8.

If input of NTD have some clauses which include variables x, y, the input that change variables y to x (and reduce all $x \wedge x \to x$, $x \wedge \overline{x} \to 0$ to become indistinguishable what variables changed) also in NTD.

$$\forall p \in NTD \left(\exists x, y \in p \, (x, y \in d \in p) \to q \in NTD \mid q = p \begin{pmatrix} \cdots & x & \overline{x} & y & \overline{y} & \cdots \\ \cdots & x & \overline{x} & x & \overline{x} & \cdots \end{pmatrix} \right)$$

$x, y \in d \in p$: DNF p have some clauses d that include variable x, y.

Proof. (Proof by contradiction.) Assume to the contrary that

$$\exists p \in NTD \left(\exists x, y \in p \, (x, y \in d \in p) \wedge q \notin NTD \mid q = p \begin{pmatrix} \cdots & x & \overline{x} & y & \overline{y} & \cdots \\ \cdots & x & \overline{x} & x & \overline{x} & \cdots \end{pmatrix} \right)$$

Because of $p \equiv \top$, it is trivial that $q \equiv \top$ and $q \begin{pmatrix} \cdots & x & \overline{x} & \cdots \\ \cdots & \overline{x} & x & \cdots \end{pmatrix} \equiv \top$. So some $q \begin{pmatrix} \cdots & x & \overline{x} & \cdots \\ \cdots & \{x, \overline{x}\} & \{x, \overline{x}\} & \cdots \end{pmatrix} \equiv \top$ from assumption $q \notin NTD$.

However,

$$p \in NTD \to p \begin{pmatrix} \cdots & x & \overline{x} & \cdots \\ \cdots & \{x, \overline{x}\} & \{x, \overline{x}\} & \cdots \end{pmatrix} \not\equiv \top, p \begin{pmatrix} \cdots & y & \overline{y} & \cdots \\ \cdots & \{y, \overline{y}\} & \{y, \overline{y}\} & \cdots \end{pmatrix} \not\equiv \top$$

So following are only tautology of changing positive / negative variables

$$p \begin{pmatrix} \cdots & x & \overline{x} & y & \overline{y} & \cdots \\ \cdots & \overline{x} & x & y & \overline{y} & \cdots \end{pmatrix} \equiv \top, p \begin{pmatrix} \cdots & x & \overline{x} & y & \overline{y} & \cdots \\ \cdots & x & \overline{x} & \overline{y} & y & \cdots \end{pmatrix} \equiv \top$$

then q satisfy following conditions.

$$q \begin{pmatrix} \cdots & x & \overline{x} & x & \overline{x} & \cdots \\ \cdots & \overline{x} & x & x & \overline{x} & \cdots \end{pmatrix} \equiv \top, q \begin{pmatrix} \cdots & x & \overline{x} & x & \overline{x} & \cdots \\ \cdots & x & \overline{x} & \overline{x} & x & \cdots \end{pmatrix} \equiv \top$$

This means that we have to treat each x, y in $q = p\begin{pmatrix} \cdots & x & \overline{x} & y & \overline{y} & \cdots \\ \cdots & x & \overline{x} & x & \overline{x} & \cdots \end{pmatrix}$ separately. That is, $q = p\begin{pmatrix} \cdots & x & \overline{x} & y & \overline{y} & \cdots \\ \cdots & x & \overline{x} & x & \overline{x} & \cdots \end{pmatrix}$ is irreducible about $x \wedge x \rightarrow x$ and $x \wedge \overline{x} \rightarrow 0$, so $\forall x, y \in p\,(x, y \notin d \in p)$. This is contradict assumption $\exists x, y \in p\,(x, y \in d \in p)$. □

Theorem 2.9.

Number of neighbor input in NTD is over polynomial number of input length.

Proof. Mentioned above 2.8, if $p \in NTD$ and exists $x, y \in c \in p$ then $q \in NTD \mid q = p\begin{pmatrix} \cdots & x & \overline{x} & y & \overline{y} & \cdots \\ \cdots & x & \overline{x} & x & \overline{x} & \cdots \end{pmatrix}$. Because of symmetry of $y, \overline{y}$ in tautology, $q' \in NTD \mid q' = p\begin{pmatrix} \cdots & x & \overline{x} & y & \overline{y} & \cdots \\ \cdots & x & \overline{x} & \overline{x} & x & \cdots \end{pmatrix}$ also true. If p does not have some clauses that include x, y in same clauses, we can change p to p' that have some clauses that include x, y in same clauses like 2.5. If generated formula q, q' consist of same clauses, we can change p to p'' that $p''\begin{pmatrix} \cdots & x & \overline{x} & y & \overline{y} & \cdots \\ \cdots & x & \overline{x} & x & \overline{x} & \cdots \end{pmatrix}$ and $p''\begin{pmatrix} \cdots & x & \overline{x} & y & \overline{y} & \cdots \\ \cdots & x & \overline{x} & \overline{x} & x & \cdots \end{pmatrix}$ do not consist of same clauses like 2.6. When we add some clauses previous changing, adding clauses include some clauses that have unique variables set that does not have another generated formuras not to come to the same clauses. In this way, we can generate at least two times of NTD from some NTD by reducing $y, \overline{y} \rightarrow x, \overline{x}$ or $y, \overline{y} \rightarrow \overline{x}, x$.

On the other hand, we can repeat above variables reducing each variables in p. To confirm number of variables type in p, we cannot limit the number no more than logarithm number of input length $|p|$. Therefore generated NTD amount to over polynomial number of input length $|p|$. □

Theorem 2.10.

NNF circuit family have to use over polynomial number of gates of input length to compute NTD.

Proof. Mentioned above 1.6, NNF have to unique gate which Different variables of neighbor input. Mentioned above 2.9, size of NTD is over polynomial size of input length. Therefore size of NNF circuit family that compute NTD is over polynomial size. □

Theorem 2.11.

$NTD \notin P$

Proof. Mentioned above 1.4, NNF circuit family can compute P problem with polynomial number of gates of input length. However mentioned above 2.10, NNF circuit family have to use over polynomial number of gates of input length to compute NTD. Therefore NTD is not in P. □

Theorem 2.12.

$P \subsetneq PH$

Proof. Mentioned above 2.7, $NTD \in PH$, but mentioned above 2.11, $NTD \notin P$. Therefore PH is not in P. □

References

[Sipser] Michael Sipser, (translation) OHTA Kazuo, TANAKA Keisuke, ABE Masayuki, UEDA Hiroki, FUJIOKA Atsushi, WATANABE Osamu, Introduction to the Theory of COMPUTATION Second Edition, 2008