CIRCUIT COMPLEXITY AND PROBLEM STRUCTURE IN HAMMING SPACE

KOJI KOBAYASHI

Abstract. This paper describes about relation between circuit complexity and accept inputs structure in Hamming space by using almost all monotone circuit that emulate deterministic Turing machine (DTM).

Circuit family that emulate DTM are almost all monotone circuit family except some NOT-gate which connect input variables (like negation normal form (NNF)). Therefore, we can analyze DTM limitation by using this NNF Circuit family.

NNF circuit have symmetry of OR-gate input line, so NNF circuit cannot identify from OR-gate output line which of OR-gate input line is 1. So NNF circuit family cannot compute sandwich structure effectively (Sandwich structure is two accept inputs that sandwich reject inputs in Hamming space). NNF circuit have to use unique AND-gate to identify each different vector of sandwich structure. That is, we can measure problem complexity by counting different vectors.

Some decision problem have characteristic in sandwich structure. Different vectors of Negate HornSAT problem are at most constant length because we can delete constant part of each negative literal in Horn clauses by using definite clauses. Therefore, number of these different vector is at most polynomial size. The other hand, we can design high complexity problem with almost perfct nonlinear (APN) function.

1. Introduction

In this paper, we consider the relation between circuit complexity and accept inputs structure in Hamming space by using almost all monotone circuit that emulate deterministic Turing machine (DTM). For example, we analyze Negation HornSAT

Date: 2018-05-29.

problem complexity and design new problem that include high complexity of primitive polynomial non linearity.

In computational complexity, we use circuit family to analyze problem complexity, and we find out some result such as $PARITY \notin AC_0$ [Ajtai, Furst], $CLIQUE \notin mP$ monotone circuit family with polynomial size [Razborov]. The purpose of this paper is to provide new approach to analyze problem complexity by corresponding problem input structure in Hamming space and gate in circuit family which emulate DTM.

2. NNF circuit family

First, we define NNF circuit family that is almost all monotone circuit. Explained in book [Sipser] Circuit Complexity section 9.30, 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 2.1.

We will use the terms;

"NNF Circuit Family" as circuit family that have no NOT-gate except connecting INPUT-gates directly (like negation normal form).

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

Figure 2.1 is example of a NNF circuit.

Theorem 2.2.

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] theorem 9.30 proof. See [Sipser] for detail.

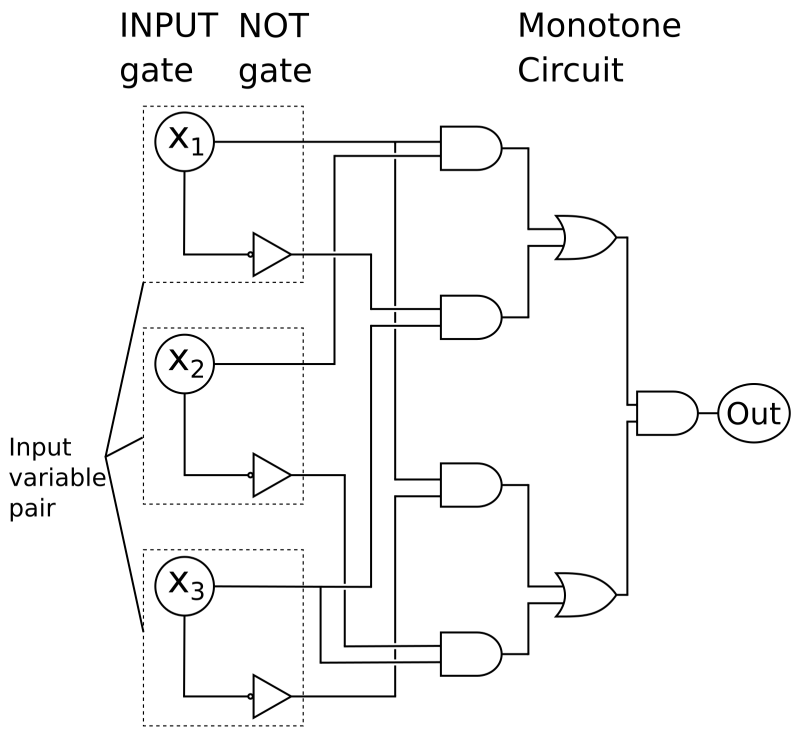

FIGURE 2.1. NNF circuit

NNF circuit family can emulate DTM by computing every step's cell values (and head state if head on the cell). Figure 2.2 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 at most side cells in each step).

Figure 2.3 shows example of $c_{u,v}$ sub circuit that transition function is "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 $(c_{u-1,v-1}, c_{u-1,v}, c_{u-1,v+1}) = (q_k 0, q_- 0, q_- 0)$. q_- means "no head on the cell".

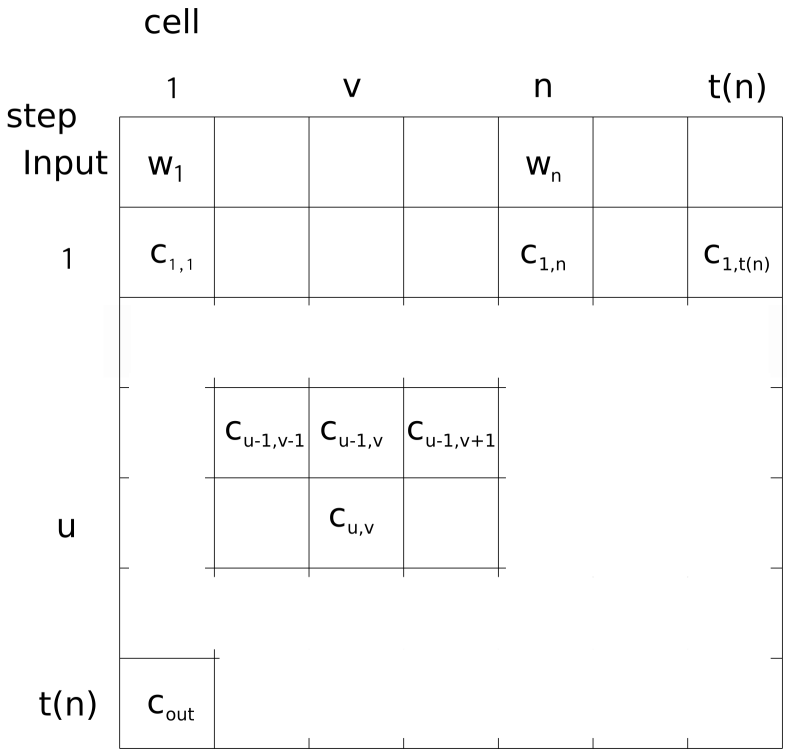

FIGURE 2.2. NNF circuit block diagram

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 $c_{u,v}$. To simplify, NNF circuit include separate three gates $\wedge_{\delta,-1}$, $\wedge_{\delta,0}$, $\wedge_{\delta,+1}$ according to head exists position $c_{u-1,v-1}$, $c_{u-1,v}$, $c_{u-1,v+1}$, and special transition function δ_- which correspond to no head transition (keep current tape value). 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, so $\wedge_\delta$ output is connected to $\vee_{w,qm}$ in $c_{u,v}$ and decide $c_{u,v}$ condition. Because DTM have constant

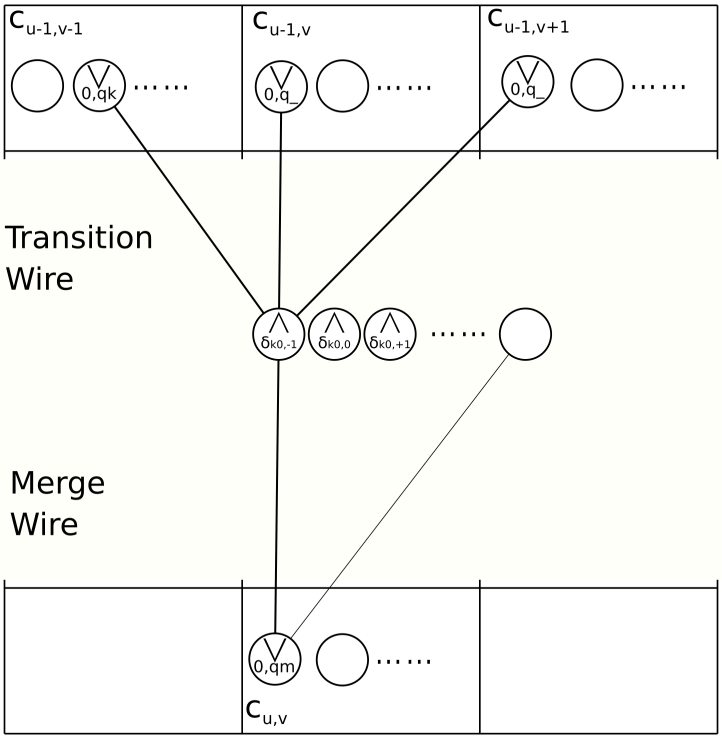

FIGURE 2.3. $c_{u,v}$ circuit

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 2.3.

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

Confirm NNF circuit family behavior. We define some term that decide relation of inputs.

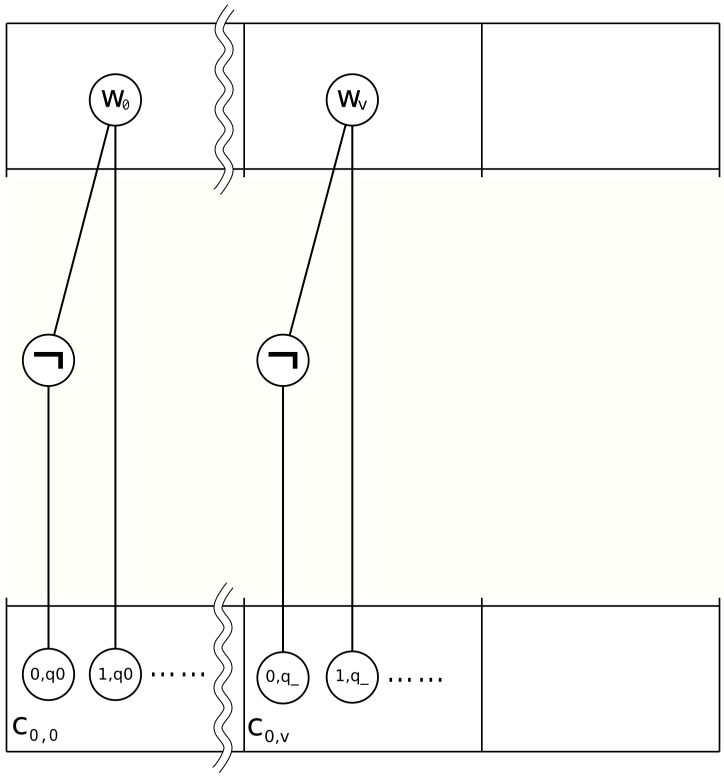

FIGURE 2.4. First step

Definition 2.4.

We will use the term;

"Neighbor input (pair)" as accept inputs pair that no accept inputs exists between these accept input in Hamming space.

"Boundary input (set) of neighbor input" as reject inputs that exist between neighbor inputs in Hamming space.

"Different variables" as all difference part of values in neighbor input pair.

"Different vector" as vector and inverse vector pair which start and end point is neighbor input pair in Hamming space. To simplify, we use $\overline{1} = -1$.

"Neighbor distance" as different vector length.

"Sandwich structure" as connected graph which nodes are accept inputs in Hamming space.

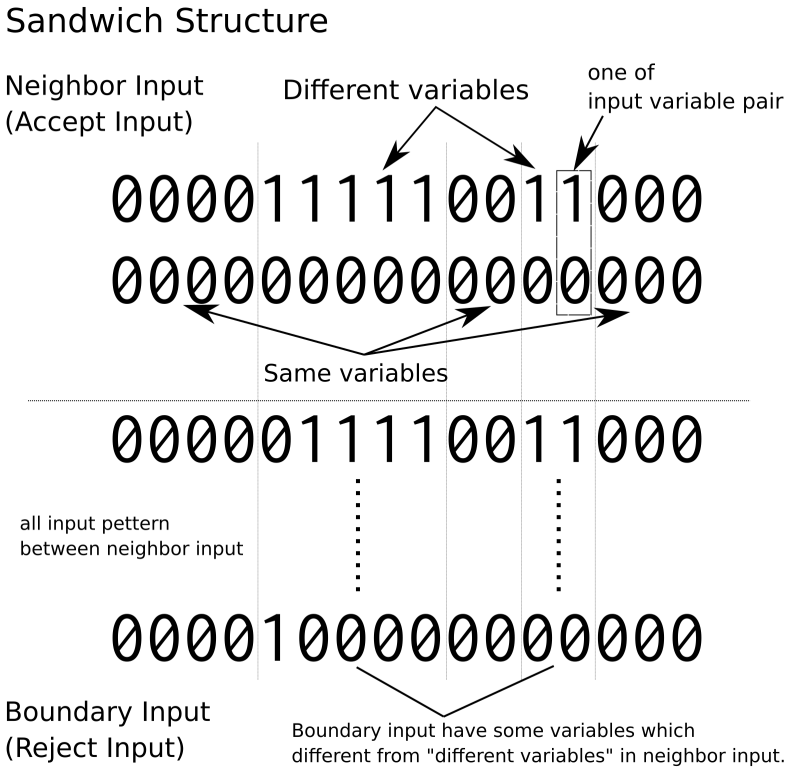

FIGURE 2.5. Sandwich structure

Figure 2.5 shows example of sandwich structure which neighbor input pair is 0000111110011000 and 0000000000000000. In this case, _____11111__11___ and _____00000__00___ are different variables, and (0000111110011000) and $\left(0000\overline{11111}00\overline{11}000\right)$ are different vector, neighbor distance is 7.

"Effective circuit of accept input t" as one of minimal sub circuit in NNF circuit that decide circuit output as 1 with accept input t. Effective circuit do not include gates which output 0, or even if these gates change output 0 and effective circuit keep output 1.

Figure 2.7 shows example of effective circuit which circuit is 2.1 and input is $\{x_1, x_2, x_3\} = \{1, 1, 0\}$. Dotted gates do not affect OUTPUT-gate even if the gate negate output, so effective circuit do not include them.

Theorem 2.5.

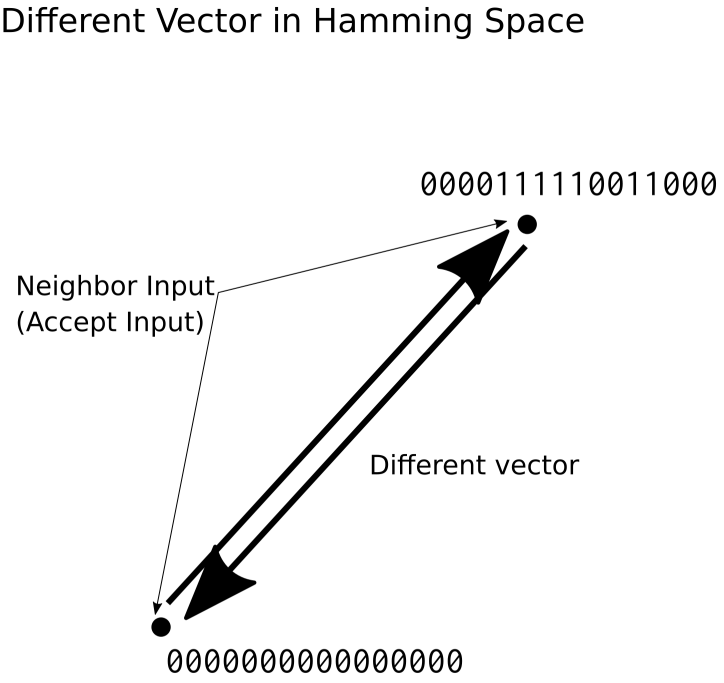

FIGURE 2.6. Different vector

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

Proof. Because all input variable pair is $\{01, 10\}$ and do not include 11 in every input. NNF circuit is almost monotone circuit, so effective circuit have to to join another accept input $\{01, 10\}$ at OR-gate to connect OUTPUT-gate. □

Figure 2.8 shows example of effective circuit which circuit is 2.1 and input are $\{x_1, x_2, x_3\} = \{1, 1, 0\}, \{0, 0, 1\}$. Effective circuit include one of input variable pair, and other side of variable pair do not become 1 in same input. So AND-gate cannot meet another effective circuit.

Theorem 2.6.

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

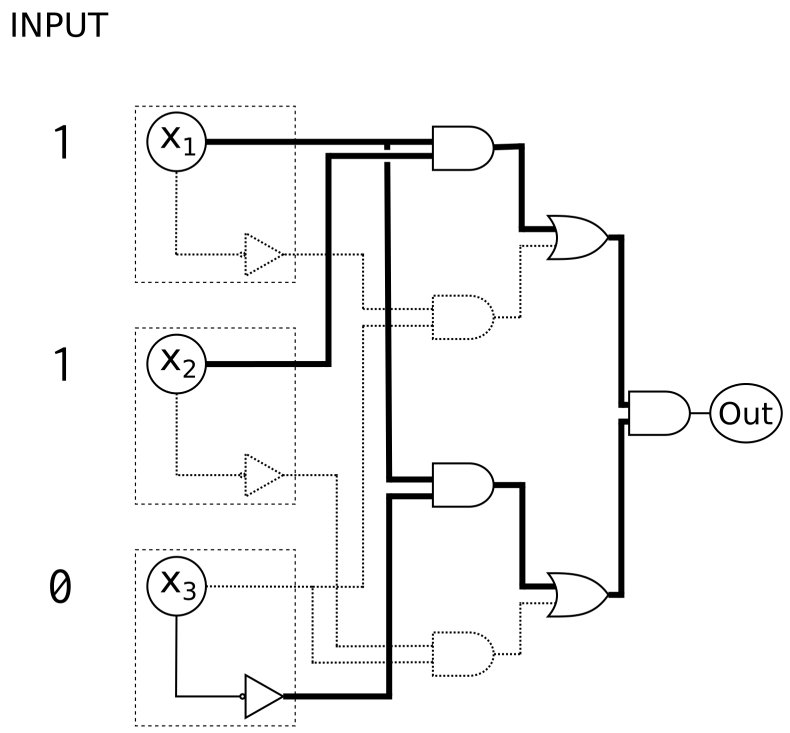

FIGURE 2.7. Effective circuit

Proof. Mentioned above 2.5, all accept input variable pair of different variables join at OR-gate. Because NNF circuit is almost all monotone circuit, there are a) b) case to join effective circuits;

a) 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. (see 2.8)

b) all different variables meet at AND-gate, and join at OR-gate after meeting AND-gate. (see 2.9)

Case a), because no boundary input become accept input, some OR-gate which join different variables become 0 if input is boundary input (2.8 1,2). 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 (2.8 3) that meet all these OR-gate which join all different variables.

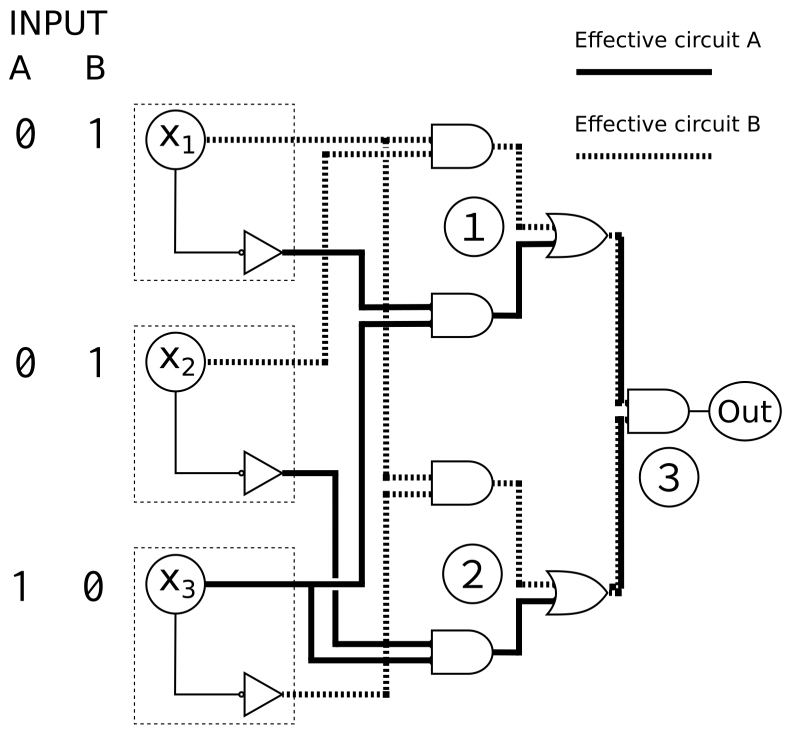

FIGURE 2.8. Different variable pair

Such AND-gate become 1 if and only if input include different variables of one side of neighbor input pair. Each pair of different variables correspond to different vector, so the AND-gate correspond to different vector.

Case b), some AND-gate become 1 if and only if input include one side of different variables. Therefore, trunk of these AND-gate (2.9 4,5) does not become 1 if input AND-gate does not include these different variables. Each pair of different variables correspond to different vector, so the AND-gate correspond to different vector.

Therefore, NNF circuit have at least one unique AND-gate that correspond to different vector to differentiate neighbor input and boundary input. □

NNF circuit can emulate DTM in polynomial size, and NNF circuit include unique AND-gate that correspond to different vector. Therefore, we can measure problem complexity by counting different vector in problem's sandwich structure.

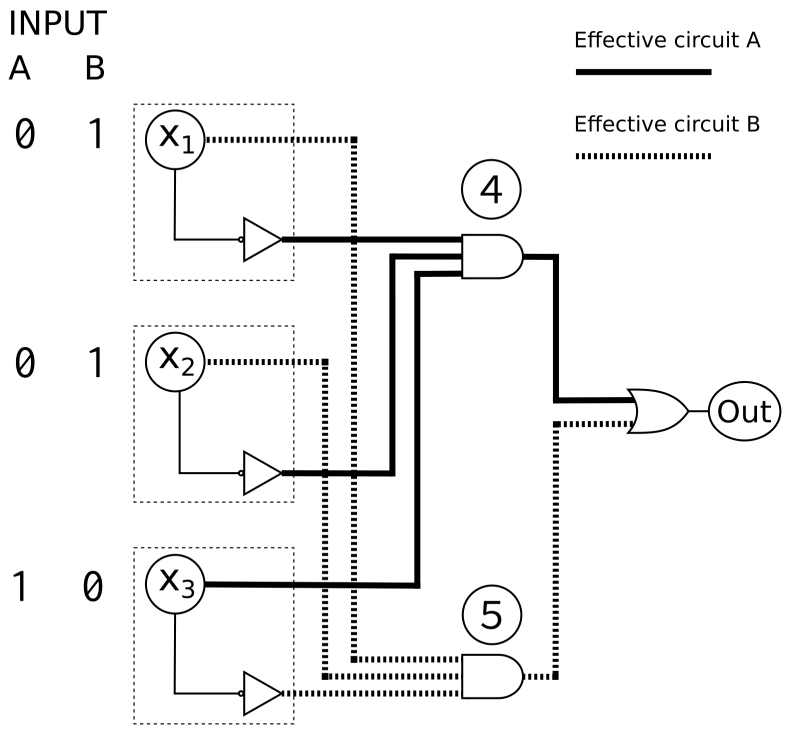

FIGURE 2.9. Example of b)

3. Negation HornSAT

Consider different vector in actual problems. Let consider Negation HornSAT problem $\overline{HornSAT}$. $\overline{HornSAT}$ can delete some negative literal which correspond definite clauses. This means that each $\overline{HornSAT}$ accept input are close each other in Hamming space. In fact, we can close neighbor distance within constant distance by devising $\overline{HornSAT}$ description.

Definition 3.1.

We will use the term $\overline{HornSAT}$ as problem if and only if Horn CNF is $\perp$.

In $\overline{HornSAT}$, we use special description as following;

x_i : Variables in $\overline{HornSAT}$. i in x_i is variable code, and x in x_i is constant code. Negative literal $\overline{x}$ is constant code which length is same as x.

$\bot_i$: Disabled Variables in $\overline{HornSAT}$ that $\bot_i = \bot$. $\bot$ in $\bot_i$ is constant code which length is same as x.

$-$: Ignored filler code in $\overline{HornSAT}$.

All another symbol $\wedge \vee ()$ are also constant length code which is same as x.

Theorem 3.2.

In $\overline{HornSAT}$, there is some sandwich structure which neighbor distance is at most constant size, and number of different vector is at most polynomial size.

Proof. Let $t = x_i \wedge (\overline{x_i} \vee \cdots) \wedge \cdots \in \overline{HornSAT}$. can reduce another $t' = x_i \wedge (\bot_i \vee \cdots) \wedge \cdots \in \overline{HornSAT}$ because we can delete all literal $\overline{x_i}$ by using definite clauses x_i. Neighbor distance between t, t' is constant because difference between $\overline{x_i}$ and $\bot_i$ is constant part of $x, \bot$. Because all $\bot_i = \bot$, we can reduce all $\bot_i \to \cdots \to \bot_- \to \bot$ by overwriting $-$ at most constant size in each steps, and each neighbor distance are at most constant. That is, we can reduce t' to $t'' = x_i \wedge (\bot \vee \cdots) \wedge \cdots \in \overline{HornSAT}$ with overwriting constant distance.

The other hand, we can apply above steps all reduction of negative literals. When some clauses have no variables like $(\bot \vee \cdots \vee \bot)$, we can overwrite any code in formula because the formula is $\bot$. Therefore, all of $\overline{HornSAT}$ have neighbor input that distance is at most constant.

Consider number of $\overline{HornSAT}$ different vectors. Let different distance is constant k. Because different distance is k, number of different vector is combination of different variables $\begin{pmatrix} n \\ k \end{pmatrix}$ and combination of variables pair in constant code 2^k.

$\begin{pmatrix} n \\ k \end{pmatrix} \times 2^k = \frac{n!}{k! \times (n-k)!} \times 2^k \leq O\left(n^k\right)$

Therefore we obtain theorem. □

4. Design high complexity problem

Consider designing high complexity problem. Mentioned in this paper, computational complexity correspond to problem structure in Hamming space, especially

number of different vector. Therefore, we can design high complexity problem by designing high cardinal number of different vector.

We use Almost perfect nonlinear function (APN function)[Dobbertin] to design high cardinal number of different vector in F_{2^n} vector space. APN function f is;

Different function: $f_a : F_{2^n} \ni x \mapsto f(x+a) - f(x) \in F_{2^n}$

and x that $f_a(x) = b$(b: constant) are at most 2.

In this case, $f_a(x) = f(x+a) - f(x)$ is different vector of $f(x+a)$ and $f(x)$, so APN function f have many different vector which number is half of x.

Definition 4.1.

We will use the term;

"$APNR$" as problem with input w, input length $|w|$, APN function f;

$w = uv$, $|u| = |v|$, $f(u) = v$

Theorem 4.2.

$APNR \in PH$

Proof. We can compute $APNR$ with constant alternating Turing Machine (ATM) by computing following way.

(1) Select minimal polynomial $g(x)$ as existence.
(2) Check polynomial solution α as following;
    Reject w if $\alpha^{2^d-1} \neq 1$.
    Select α^r $(0 < r < 2^d - 1)$ as universal, and reject $g(x)$ if $\alpha^r = 1$
(3) Construct ANR function $f(x)$ by using $g(x)$ and APN power functions, and if $f(u) = v$ then accept w.

We can compute above procedure in polynomial step with constant alternation. So $APNR \in PH$. □

Theorem 4.3.

$APNR$ have $2^{|u|-1}$ number of different vector.

Proof. $APNR$ different vector $w_p + w_q$($w_p = u_p v_p$,$w_q = u_q v_q$) is;

$w_p + w_q = (u_p + u_q)(v_p + v_q)$

$= (u_p + u_q)(f(u_p) + f(u_q))$

Because APN function definition, $u_p + u_q = a$ then there are at most 2 (u_p, u_q) that $f(u_p) + f(u_q) = b$. That is, even if different vector u_p, u_q become same, there are many different vector $f(u_p) + f(u_q)$ which depend on $u_p = u_q + a$ value.

Therefore, $APNR$ have many different vector that depend on u value, and number of $APNR$ different vector is $2^{|u|-1}$. □

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

[Ajtai] Ajtai, M. Komlós, J. and Szemerédi, E.: An 0(n log n) sorting network, STOC(1983).

[Furst] Furst, M. Saxe, J.B. and Sipser, M.: Parity, circuits, and the polynomial-time hierarchy Mathematical Systems Theory(1984)

[Razborov] Razborov, A.: Lower bounds on the monotone complexity of some Boolean functions. Mathematics of the USSR(1985)

[Dobbertin] Dobbertin, H.: H. Dobbertin, Almost perfect nonlinear functions on $GF(2^n)$: the Niho case, Information and Computation 151, (1999).