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 family cannot compute sandwich structure effectively. Sandwich structure is two accept inputs that sandwich reject inputs in Hamming space. So 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 dicision problem have caracteristic 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, special subset of Negate CNFSAT problem have different vector which number is over polynomial size.

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), and confirm Negation HornSAT problem and Negation CNFSAT problem. In computational complexity, we use circuit family to analyze problem complexity, and we find out some result such as

Date: 2018-04-05.

$PARITY \notin AC_0$ [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, 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] proof. See [Sipser] for detail.

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

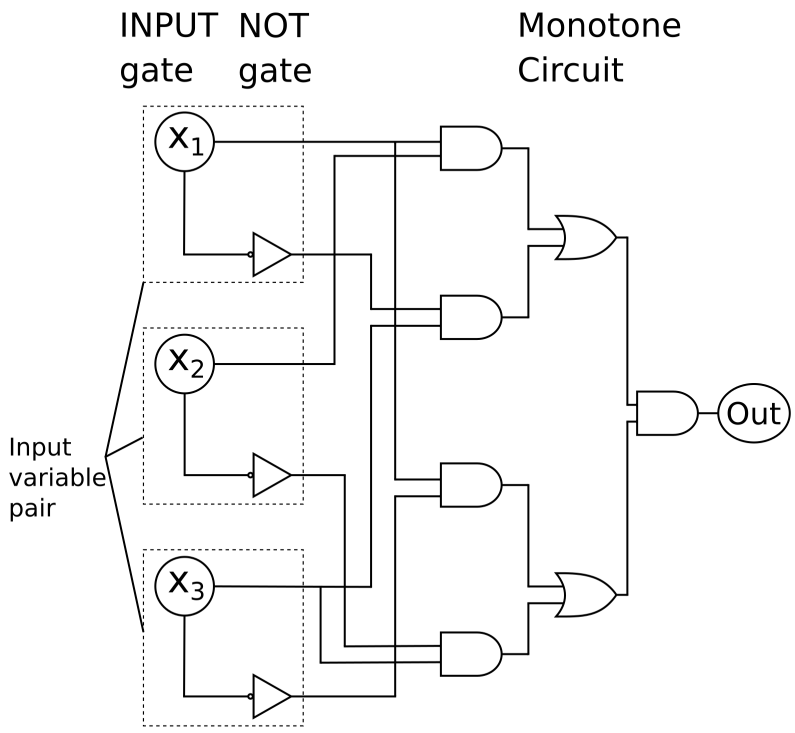

FIGURE 2.1. NNF circuit

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".

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

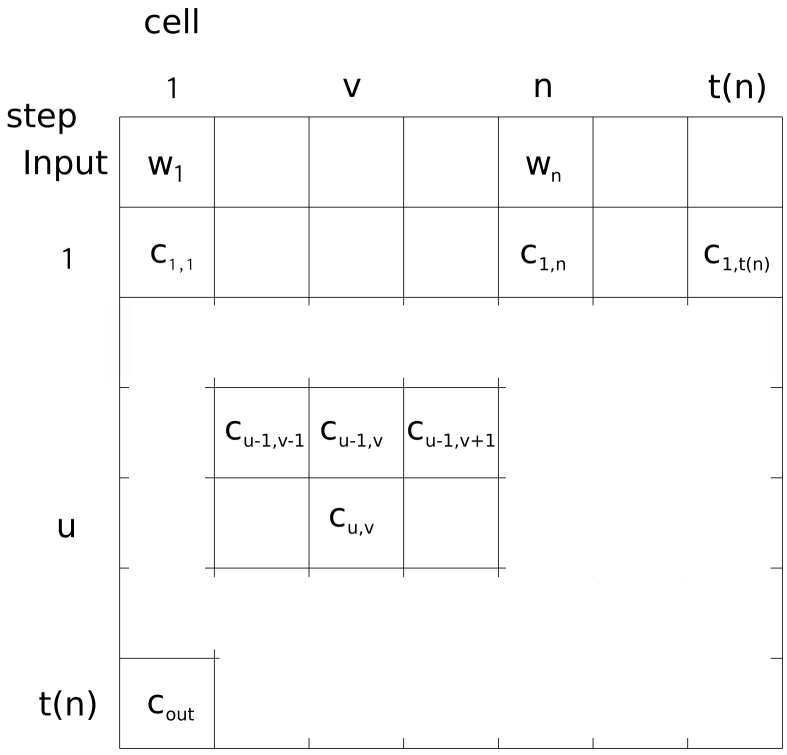

FIGURE 2.2. NNF circuit block diagram

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 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.

□

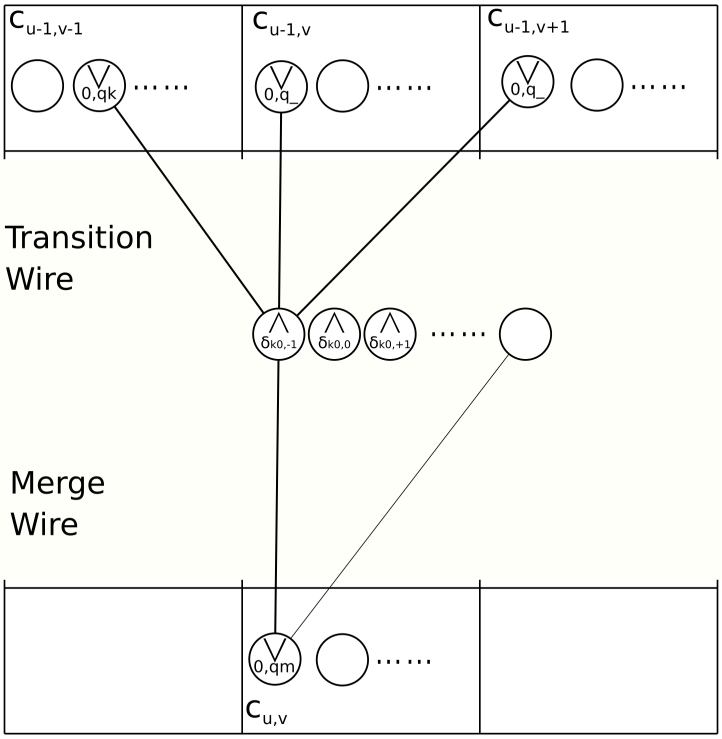

FIGURE 2.3. $c_{u,v}$ circuit

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.

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.

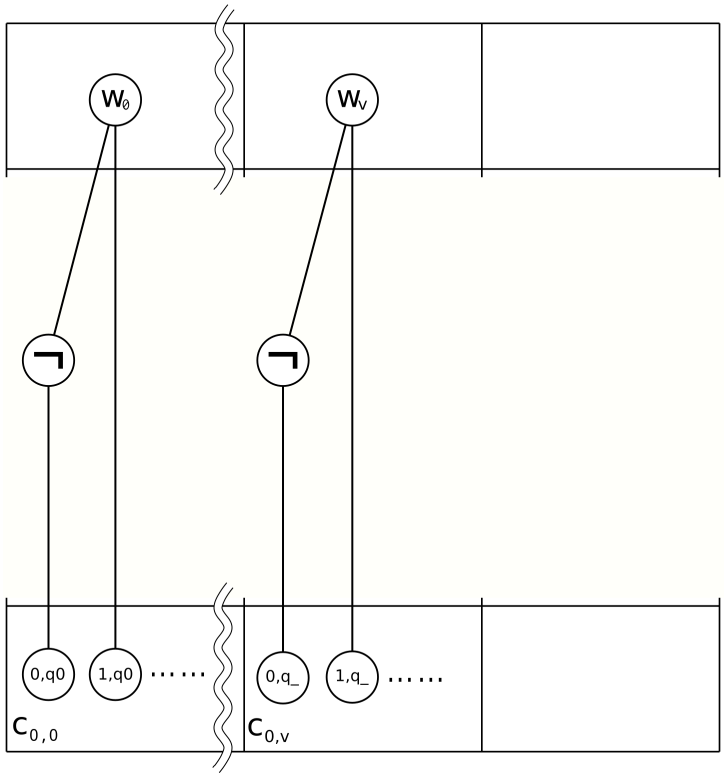

FIGURE 2.4. First step

"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 shows example of sandwich structure which neighbor input pair is 0000111110011000 and 0000000000000000. In this case, $\circ\circ\circ\circ 11111\circ\circ 11\circ\circ\circ$ and $\circ\circ\circ\circ 00000\circ\circ 00\circ\circ\circ$ 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

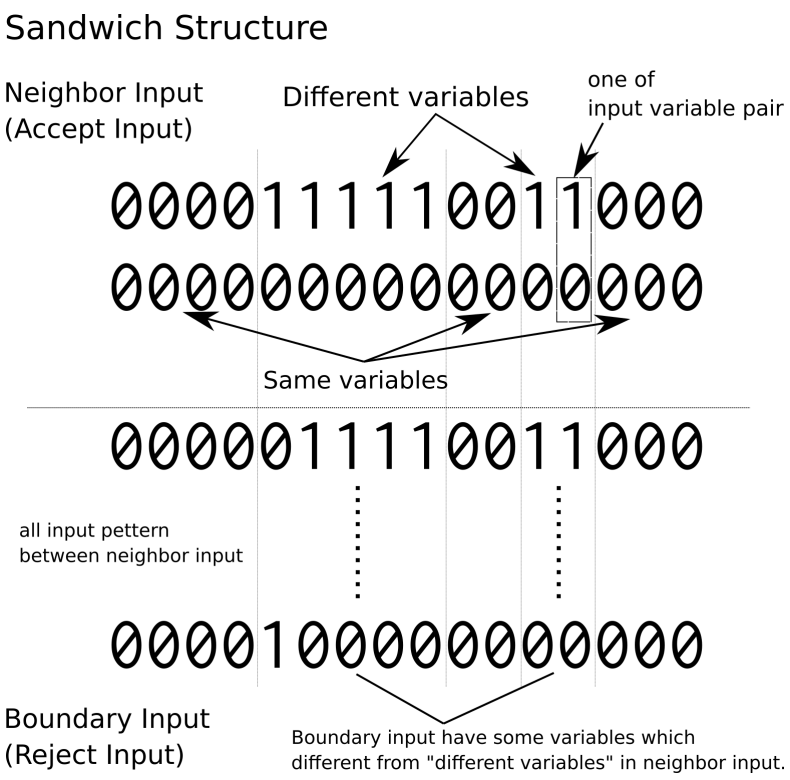

FIGURE 2.5. Sandwich structure

gates which output 0, or even if gate 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.

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,

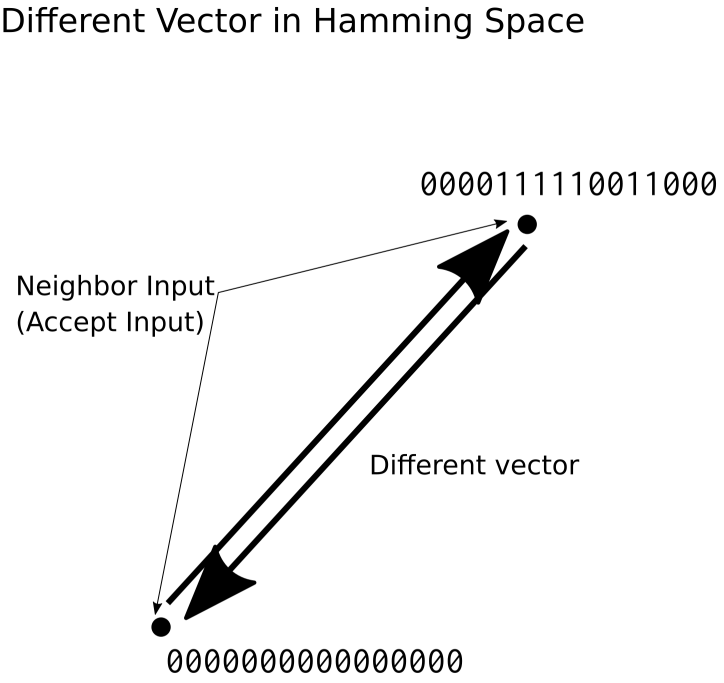

FIGURE 2.6. Different vector

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.

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)

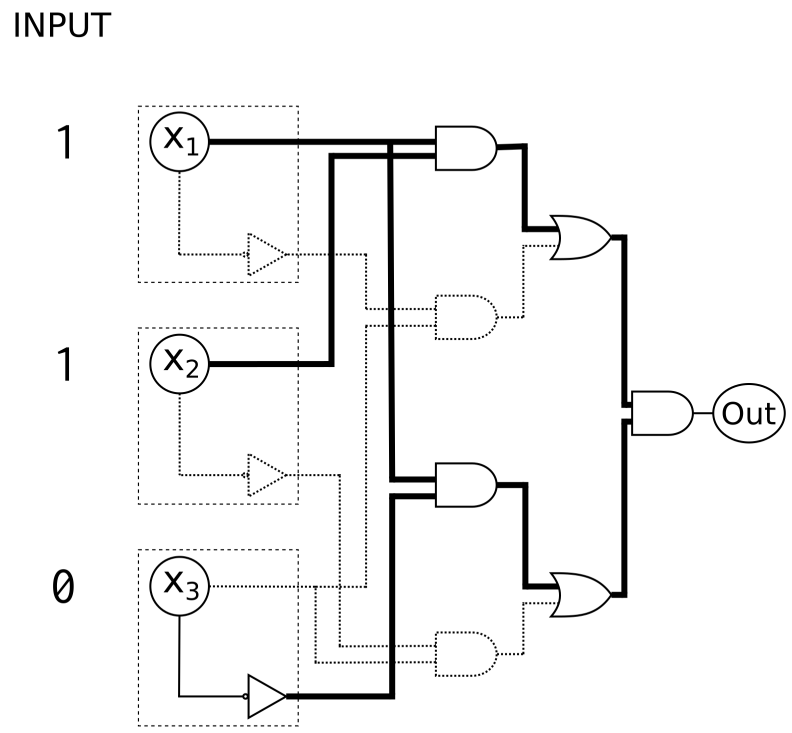

FIGURE 2.7. Effective circuit

Case a), because no boundary input become accept input, some OR-gate which join different variables become 0 if input is 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 all different variables like 2.8. 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 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.

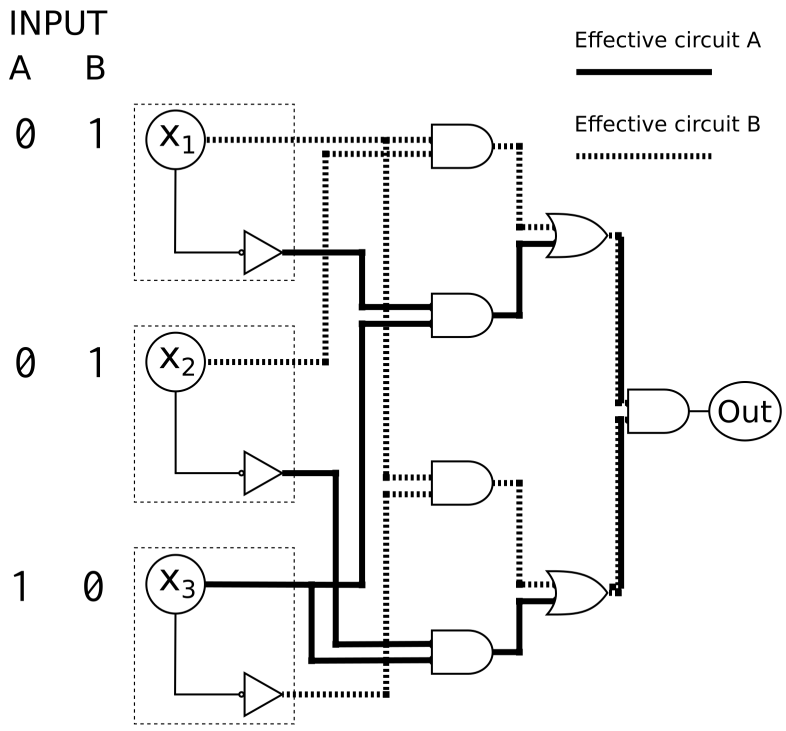

FIGURE 2.8. Different variable pair

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.

3. Negation HornSAT

Consider different vector in actual problems. Let consider Negation HornSAT problem $\overline{HornSAT}$. $\overline{HornSAT}$ can delete som 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.

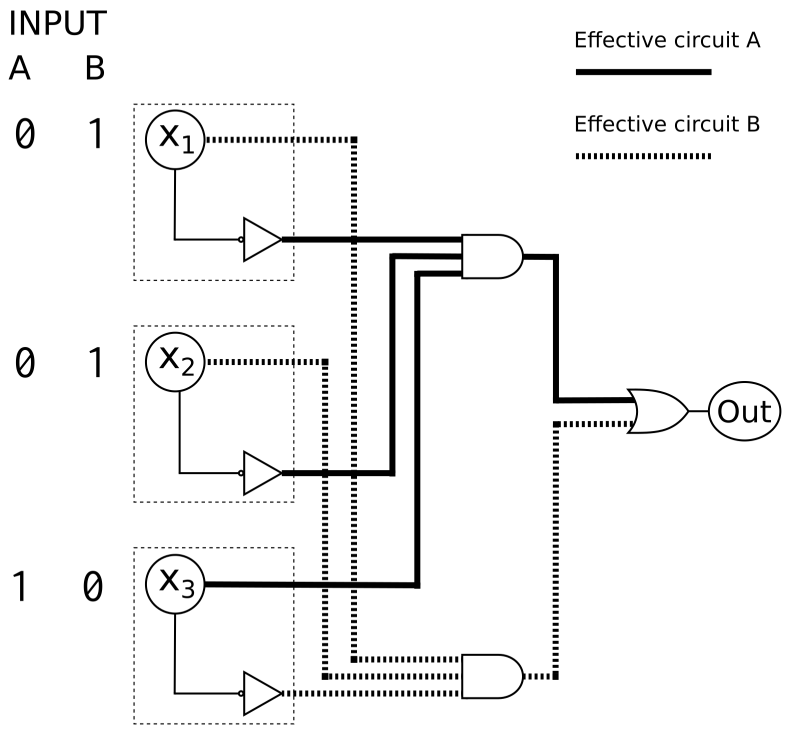

FIGURE 2.9. Example of b)

Definition 3.1.

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

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 atmost constant size, and number of different vector is atmost 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 formule 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 conbination 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. Negation CNFSAT

Consider Negation CNFSAT problem $\overline{CNFSAT}$. $\overline{CNFSAT}$ dose not have definite clauses, so we cannot delete negative literal like $\overline{HornSAT}$. However, $\overline{CNFSAT}$ is symmetry that permutate truth value assignment, so $\overline{CNFSAT}$ is symmetry that permutate all literal of one variables. We define special $\overline{CNFSAT}$ by using this literal symmetry, and analyze lower limit of different vector number.

Definition 4.1.

We will use the tenrm;

"$\overline{CNFSAT}$" as problem if and only if CNF is $\bot$.

$\overline{CNFSAT} = \{f \mid f \in CNF, f = \bot\}$

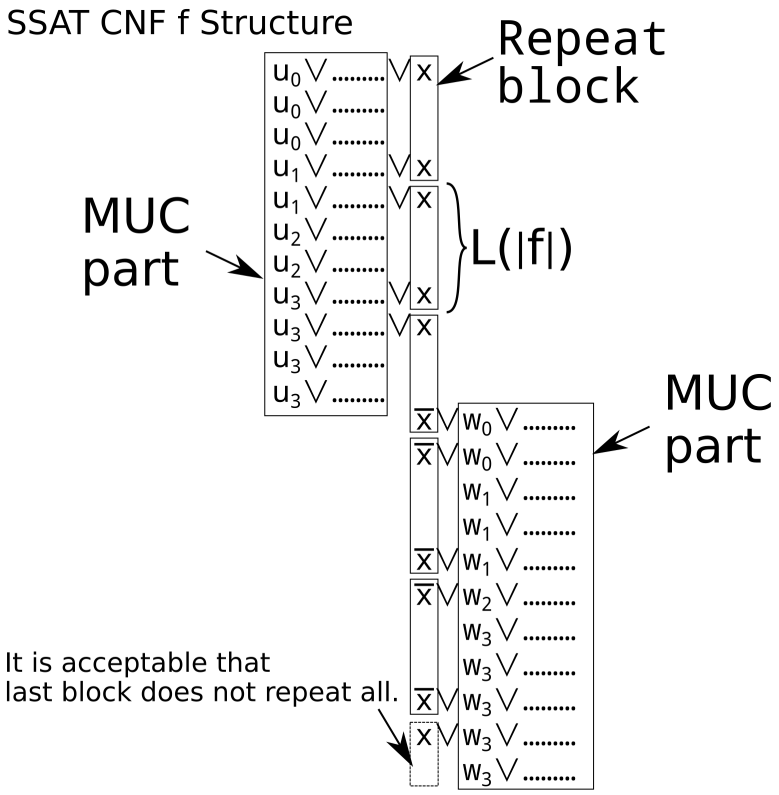

FIGURE 4.1. Example of SSAT clauses

"MUC" (Minimal Unsatisfiable CNF) as subset of $\overline{CNFSAT}$ that any CNF which delete some clauses does not become $\overline{CNFSAT}$. That is;

$MUC = \left\{ f \mid f \in \overline{CNFSAT}, h \subsetneq f \rightarrow h \notin \overline{CNFSAT} \right\}$

"$\overline{SSAT}$" as subset of $f \in MUC$ that;

If variables x delete from f and f become two different MUC,

$f \setminus x = g \wedge h \mid g, h \in MUC$

then x appear periodically when clauses sort in dictionary order. This period $L\left(|f|\right)$ satisfy

$2^{L(|f|)} > O\left(|f|^c\right)$ and $\frac{|f|}{L(|f|)} > L\left(|f|\right)$

($L\left(|f|\right)$ example is $\left(\log |f|\right)^k$, $L\left(|f|\right) = \lceil F \rceil \mid F^F = |f|$)

Figure 4.1 shows example of $f \in \overline{SSAT}$.

We use special description $\overline{CNFSAT}$ like as $\overline{HornSAT}$, and all of positive / negative literal described with flag code. That is, we can change literal positive / negative by overwriting constant code. To symplify, $\overline{CNFSAT}$ does not include same clauses. We treat CNF as set of clauses, and also treat clause as set of literals.

"$f\begin{pmatrix} x \\ y \end{pmatrix}$" as permutate x to y in f.

"$f\begin{pmatrix} x & \overline{x} \\ \overline{x} & x \end{pmatrix}$" as permutate all literal $x, \overline{x}$ to $\overline{x}, x$ in f.

"$f\begin{pmatrix} x & \overline{x} \\ x, \overline{x} & \overline{x}, x \end{pmatrix}$" as proper partical permutate $x, \overline{x}$ to $\overline{x}, x$ in f. ($f\begin{pmatrix} x & \overline{x} \\ x, \overline{x} & \overline{x}, x \end{pmatrix}$ do not decide specific permutation. So $f\begin{pmatrix} x & \overline{x} \\ x, \overline{x} & \overline{x}, x \end{pmatrix}$ means several partical permutation.)

"$[f|\, x$" as $f\begin{pmatrix} \top \\ \top, x \end{pmatrix}$ that add positive free literal x in some f clause(s).

"$x\,|g]$" as $[g|\, \overline{x}$ that add negative free literal $\overline{x}$ in some g clause(s).

"$[f|\, x\,|g]$" as $[f|\, x \wedge x\,|g]$.

However, each variables in $[f|\, x$,$x\,|g]$,$[f|\, x\,|g]$ do not bind another variables, we apply alpha equivalence at f, g.

"$f\,(x = \top)$", "$f\,(x = \bot)$" as formula that apply $x = \top, \bot$ in f.

Figure 4.2 shows example of $[f|\, x\,|g]$ clauses.

Theorem 4.2.

$$\forall f \in \overline{CNFSAT}, x \in f\left(f\begin{pmatrix} x & \overline{x} \\ \overline{x} & x \end{pmatrix} \in \overline{CNFSAT}\right)$$

Proof. It is trivial because $\overline{CNFSAT}$ is symmetric with permutation of truth value assignment. □

Theorem 4.3.

$[f|\, x\,(x = \bot) = f$, $[f|\, x\,(x = \top) \subsetneq f$

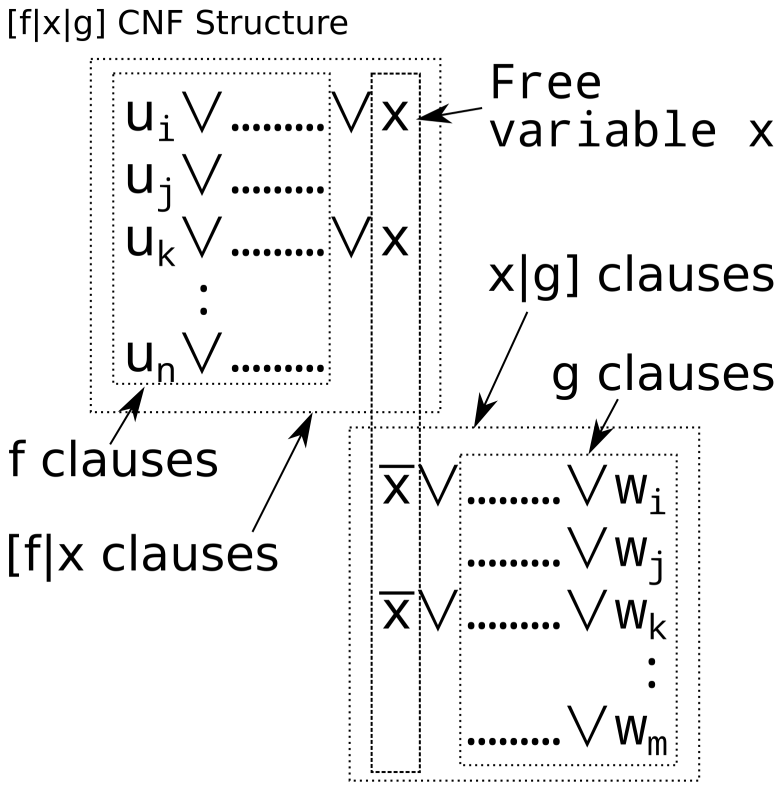

FIGURE 4.2. Example of connect clauses

$x\,|g]\,(x = \bot) \subsetneq g$, $x\,|g]\,(x = \top) = g$

Proof. It is trivial that binding relation of adding literal $x, \overline{x}$ in $[f|\,x, x\,|g]$. □

Theorem 4.4.

$\forall f, g \in \overline{CNFSAT}\,\left([f|\,x\,|g] \in \overline{CNFSAT}\right)$

$$\forall f, g \in \overline{CNFSAT}\left([f|\,x\,|g]\begin{pmatrix} x & \overline{x} \\ \overline{x} & x \end{pmatrix} \in \overline{CNFSAT}\right)$$

Proof. Mentioned above 4.3,

$f \subset [f|\,x\,|g]\,(x = \bot) \in \overline{CNFSAT}$

$g \subset [f|\,x\,|g]\,(x = \top) \in \overline{CNFSAT}$

Therefore $[f|\,x\,|g] \in \overline{CNFSAT}$.

Mentioned above 4.2,

$$\forall [f|\, x\,|g] \in \overline{CNFSAT} \left([f|\, x\,|g] \begin{pmatrix} x & \overline{x} \\ \overline{x} & x \end{pmatrix} \in \overline{CNFSAT} \right)$$

Therefore we obtain theorem. □

Theorem 4.5.

$$\forall f, g \in MUC \left([f|\, x\,|g] \begin{pmatrix} x & \overline{x} \\ y, \overline{y} & \overline{y}, y \end{pmatrix} \notin \overline{CNFSAT} \right)$$

Proof. Mentioned above 4.1 4.2, (considering symmetry of $y, \overline{y}$) $[f|\, x\,|g] \begin{pmatrix} x & \overline{x} \\ y, \overline{y} & \overline{y}, y \end{pmatrix}$ is 3 cases that;

$$[f|\, x \begin{pmatrix} x \\ y, \overline{y} \end{pmatrix} \wedge x\,|g] \begin{pmatrix} \overline{x} \\ y, \overline{y} \end{pmatrix}$$

$$\left([f|\, x \begin{pmatrix} x \\ y, \overline{y} \end{pmatrix} \right) \wedge x\,|g] \begin{pmatrix} \overline{x} \\ \overline{y} \end{pmatrix}$$

$$[f|\, x \begin{pmatrix} x \\ y \end{pmatrix} \wedge \left(x\,|g] \begin{pmatrix} \overline{x} \\ \overline{y}, y \end{pmatrix} \right)$$

In $[f|\, x \begin{pmatrix} x \\ y, \overline{y} \end{pmatrix} \wedge x\,|g] \begin{pmatrix} \overline{x} \\ y, \overline{y} \end{pmatrix}$ case, if $y = \top$ then

$$\left([f|\, x \begin{pmatrix} x \\ y, \overline{y} \end{pmatrix} \wedge x\,|g] \begin{pmatrix} \overline{x} \\ y, \overline{y} \end{pmatrix} \right) (y = \top)$$

$$= [f|\, x \begin{pmatrix} x \\ y, \overline{y} \end{pmatrix} (y = \top) \wedge x\,|g] \begin{pmatrix} \overline{x} \\ y, \overline{y} \end{pmatrix} (y = \top)$$

Some clauses in $[f|\, x \begin{pmatrix} x \\ y, \overline{y} \end{pmatrix}$ include literal y, then

$$[f|\, x \begin{pmatrix} x \\ y, \overline{y} \end{pmatrix} (y = \top) \subsetneq f$$

So

$$f' = f \setminus [f|\, x \begin{pmatrix} x \\ y, \overline{y} \end{pmatrix} (y = \top) \neq \emptyset$$

then

$$[f|\, x \begin{pmatrix} x \\ y, \overline{y} \end{pmatrix} (y = \top) = f \setminus f' \subsetneq f$$

and also $x\,|g] \begin{pmatrix} \overline{x} \\ y, \overline{y} \end{pmatrix} (y = \top)$

$$x\,|g] \begin{pmatrix} \overline{x} \\ y, \overline{y} \end{pmatrix} (y = \top) = g \setminus g' \subsetneq g$$

Because $f, g \in MUC$ and f, g do not have same variables, there are some truth value assignment t that;

$$\left([f|\, x \begin{pmatrix} x \\ y, \overline{y} \end{pmatrix} \wedge x\,|g] \begin{pmatrix} \overline{x} \\ y, \overline{y} \end{pmatrix} \right) (y = \top)\,(t)$$
$$= (f \setminus f' \wedge g \setminus g')\,(t) = \top$$

That is

$$[f|\, x \begin{pmatrix} x \\ y, \overline{y} \end{pmatrix} \wedge x\,|g] \begin{pmatrix} \overline{x} \\ y, \overline{y} \end{pmatrix} \notin \overline{CNFSAT}$$

Another case

$$\left([f|\, x \begin{pmatrix} x \\ y, \overline{y} \end{pmatrix} \right) \wedge x\,|g] \begin{pmatrix} \overline{x} \\ \overline{y} \end{pmatrix}$$

$$[f|\, x \begin{pmatrix} x \\ y \end{pmatrix} \wedge \left(x\,|g] \begin{pmatrix} \overline{x} \\ \overline{y}, y \end{pmatrix} \right)$$

also

$$\left([f|\, x \begin{pmatrix} x \\ y, \overline{y} \end{pmatrix} \right) \wedge x\,|g] \begin{pmatrix} \overline{x} \\ \overline{y} \end{pmatrix} (y = \bot)$$

$$[f|\, x \begin{pmatrix} x \\ y \end{pmatrix} \wedge \left(x\,|g] \begin{pmatrix} \overline{x} \\ \overline{y}, y \end{pmatrix} \right) (y = \top)$$

and

$$\left([f|\, x \begin{pmatrix} x \\ y, \overline{y} \end{pmatrix} \right) \wedge x\,|g] \begin{pmatrix} \overline{x} \\ \overline{y} \end{pmatrix} \notin \overline{CNFSAT}$$

$$[f|\, x \begin{pmatrix} x \\ y \end{pmatrix} \wedge \left(x\,|g] \begin{pmatrix} \overline{x} \\ \overline{y}, y \end{pmatrix} \right) \notin \overline{CNFSAT}$$

Therefore we obtain theorem. □

Theorem 4.6.

If $f, g \in MUC$, then neighbor distance of $[f|\,x\,|g] \in \overline{SSAT}$ is at least $O\left(\log\left(|[f|\,x\,|g]|\right)\right)$.

Proof. Each of $[f|\,x\,|g]$ is same code at f, g. $[f|\,x\,|g]$ are different at x existence and literal $x, \overline{x}$. Therefore, different variables of $[f|\,x\,|g] \in \overline{SSAT}$ are permutation of literal $x, \overline{x}$, or existence of variables x.

Consider permutation of literal $x, \overline{x}$. Mentioned above 4.4, if $f, g \in \overline{CNFSAT}$ then $[f|\,x\,|g]\begin{pmatrix} x & \overline{x} \\ \overline{x} & x \end{pmatrix} \in \overline{CNFSAT}$. Mentioned above 4.5, if $f, g \in MUC$ then $[f|\,x\,|g]\begin{pmatrix} x & \overline{x} \\ x, \overline{x} & \overline{x}, x \end{pmatrix} \notin \overline{CNFSAT}$. Therefore, in case of permutation of literal $x, \overline{x}$, we have to permutate all literal $x, \overline{x}$ in $[f|\,x\,|g]$. Because $[f|\,x\,|g] \in \overline{SSAT}$, literal $x, \overline{x}$ exists at least one of each period $L\left(|[f|\,x\,|g]|\right)$. So number of literal $x, \overline{x}$ is $\frac{|[f|x|g]|}{L(|[f|x|g]|)}$ and this is neighbor distance between permutation of literal $x, \overline{x}$.

Consider existence of variables x. Because $[f|\,x\,|g] \in \overline{SSAT}$, then period of variables x existence is $L\left(|[f|\,x\,|g]|\right)$ and repeat $\frac{|[f|x|g]|}{L(|[f|x|g]|)}$ times. Therefore, neighbor distance of changing one variables x is at least $\frac{|[f|x|g]|}{L(|[f|x|g]|)}$.

Because $\overline{SSAT}$ conditions,

$2^{\frac{|[f|x|g]|}{L(|[f|x|g]|)}} > 2^{L(|[f|x|g]|)} > O\left(|[f|\,x\,|g]|^c\right)$

$\frac{|[f|x|g]|}{L(|[f|x|g]|)} > L\left(|[f|\,x\,|g]|\right) > O\left(\log\left(|[f|\,x\,|g]|\right)\right)$

So neighbor distance of $[f|\,x\,|g] \in \overline{SSAT}$ is at least $O\left(\log\left(|[f|\,x\,|g]|\right)\right)$. Therefore we obtain theorem. □

Theorem 4.7.

In $\overline{SSAT}$, there is some sandwich structure which number of different vector is over polynomial size.

Proof. Mentioned above 4.6, We can make $[f|\,x\,|g] \in \overline{SSAT}$ that neighbor distance is $L\left(|[f|\,x\,|g]|\right) > O\left(\log\left(|[f|\,x\,|g]|\right)\right)$ by using $f, g \in MUC$. Number of existence of variable x in period $L\left(|[f|\,x\,|g]|\right) > O\left(\log\left(|[f|\,x\,|g]|\right)\right)$ is;

$2^{L(|[f|x|g]|)} > 2^{O(\log(|[f|x|g]|))} = O\left(|[f|\,x\,|g]|^c\right)$

Each size of different vector is at least $O(\log(|[f|\,x\,|g]|))$, and we cannot resolve these different vector to smaller vectors. Therefore, number of different vector in $[f|\,x\,|g]$ is over polynomial size, and we obtain theorem. □

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)