# ALMOST ALL MONOTONE CIRCUIT FAMILY

KOJI KOBAYASHI

Abstract. This paper describes about "Almost all monotone circuit family" and introduce its interesting advantages to measuring problem complexity.

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)).

This "NNF Circuit family" have good characteristic which present each accept input exclusivity and symmetry. Each input make some INPUT-gate and NOT-gate output 1 which set is different from another input, and meet these output in OR-gate and finally connect (specified) OUTPUT-gate. That is, NNF circuit start accept input that exclusive each other, and meet each input as symmetry input step by step and finally goal same output. Especially, some different variables which sandwich reject inputs correspond to unique AND-gate. That is, we can measure problem complexity by using different variables of accept inputs. For examples, such number of different variables type of negation HornSAT is at most polynomial size. This is one of reason that we can compute P problem easily.

## 1. NNF circuit family

Explained in [Sipser] Circuit Complexity section (Theorem 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 1.1.**

We will use the following basic terms;

*Date*: 2018-03-26.

"NNF" as Negation Normal Form.

"DTM" as Deterministic Turing Machine.

"DNF" as Disjunctive Normal Form.

"CNF" as Conjunctive Normal Form.

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

To simplify, we use symbol _ as input filler. We can ignore _ to compute input.

**Definition 1.2.**

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 1.1 is example of a NNF circuit. NNF circuit consist two subcirtuit (NOT-gate subcircuit and Monotone subcircuit).

**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 (Theorem 9.30).

NNF circuit family can emulate DTM by computing every step's cell values (and head state if head on the cell). Figure 1.2 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 1.3 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

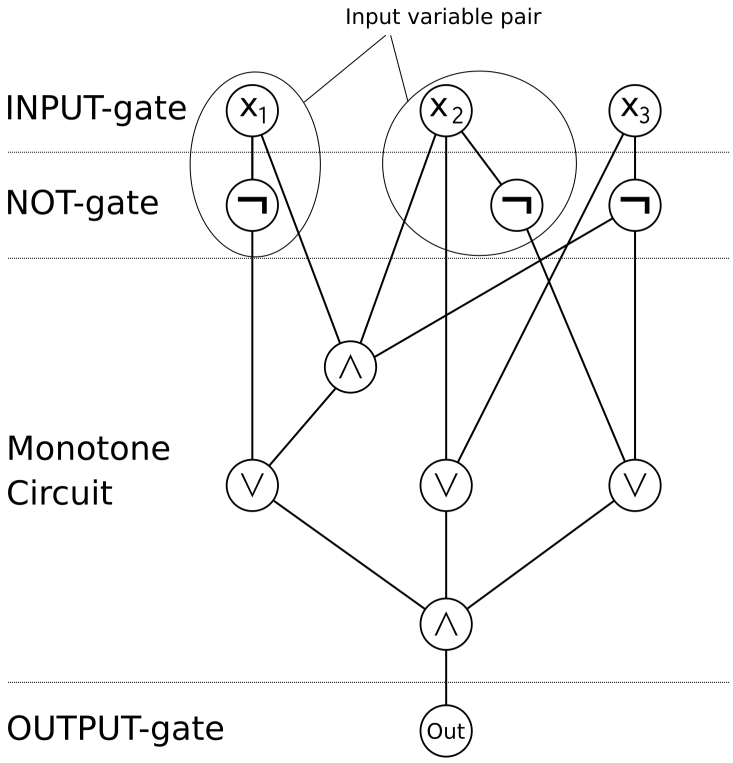


FIGURE 1.1. NNF circuit

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 $\delta$, 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 $\delta_-$ 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 $\delta$ condition. Each

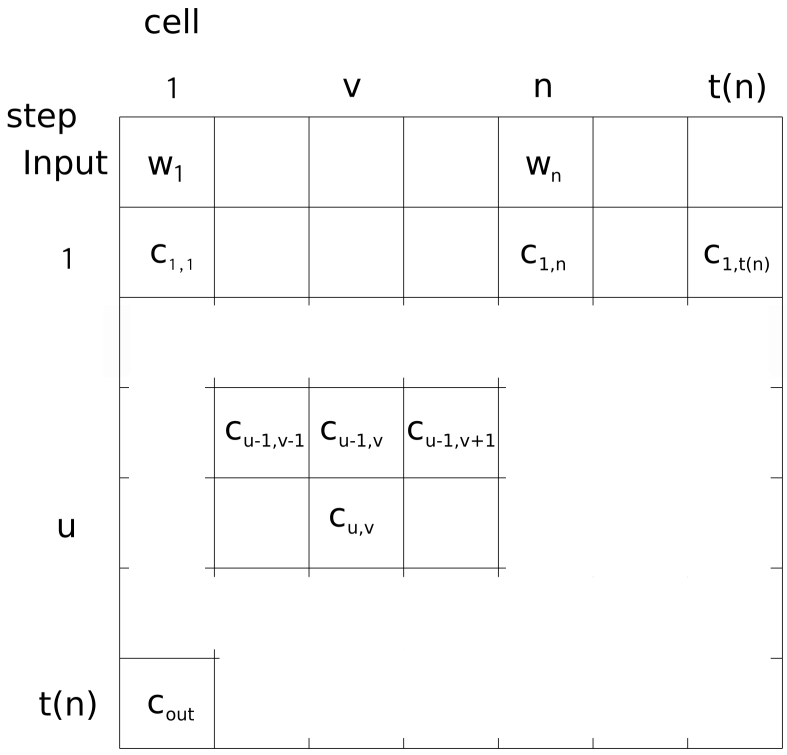


FIGURE 1.2. NNF circuit block diagram

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.

□

**Corollary 1.4.**

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

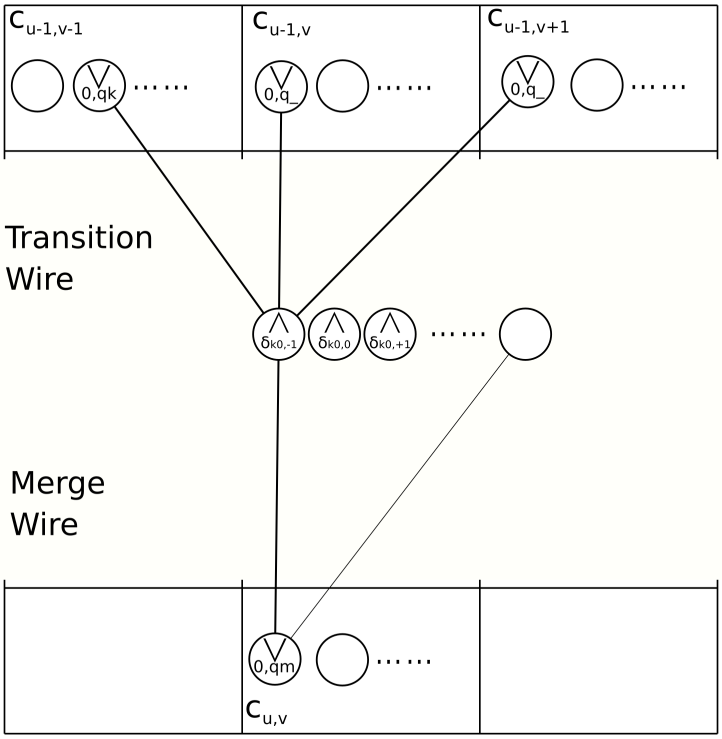


FIGURE 1.3. $c_{u,v}$ circuit

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.

**Definition 1.5.**

We will use the term;

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

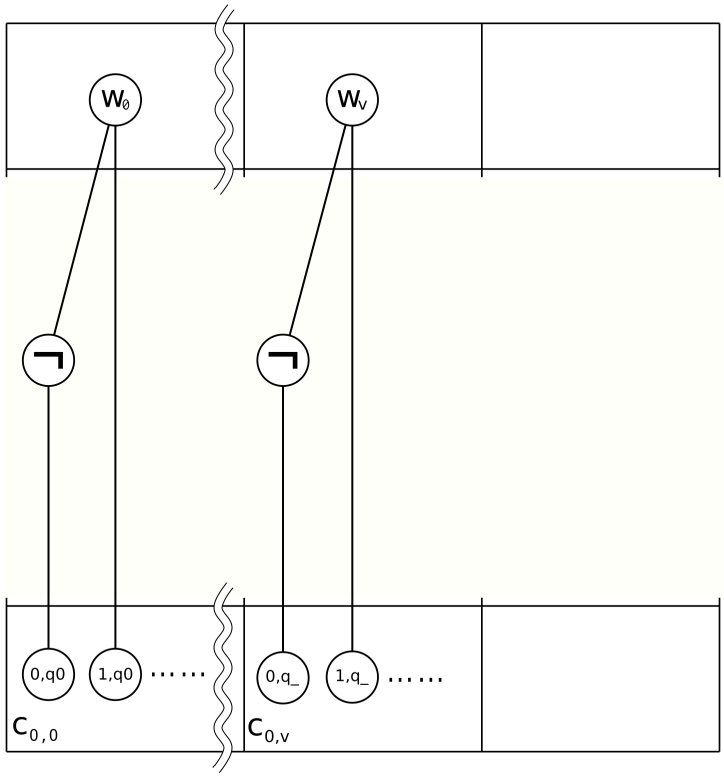


FIGURE 1.4. First step

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

"Different Variables" as all difference part of neighbor input pair.

"Same Variables" as all same part of neighbor input pair.

"Sandwich structure" as connected graph which nodes are accept inputs and edges are some of neighbor input pair in Hamming space.

Figure 1.5 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 $0000\circ\circ\circ\circ\circ 00\circ\circ 000$ are same variables.

"Effective circuit of accept input $t$" as one of minimal sub circuit of NNF circuit that decide circuit output as 1 with accept input $t$. Effective circuit do not include

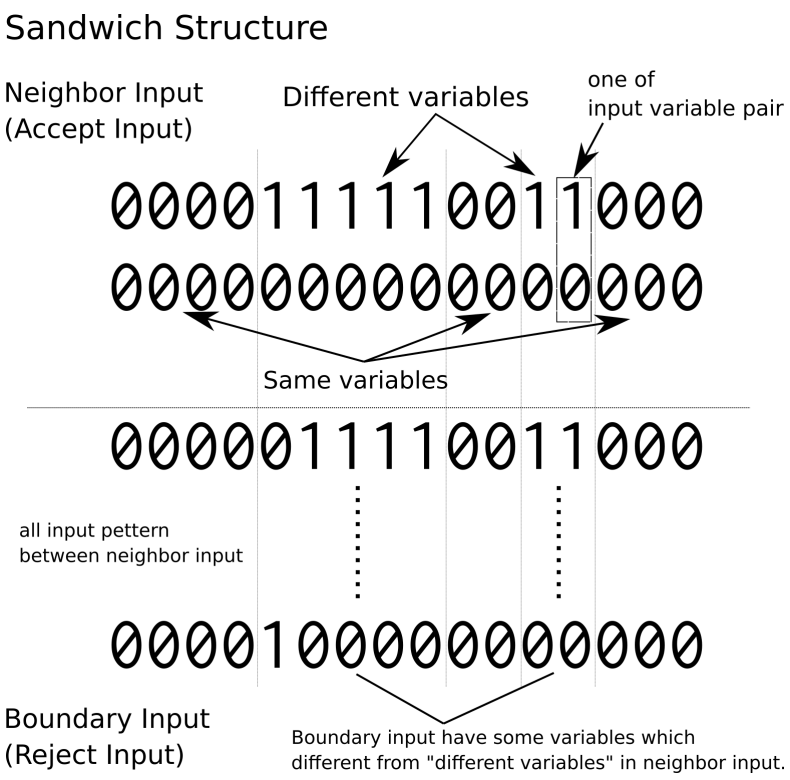


FIGURE 1.5. Sandwich structure

gates which output 0, or even if gate change output 0 and effective circuit keep output 1.

Figure 1.6 shows example of effective circuit which previous 1.1 and input $\{x_1, x_2, x_3\} = \{1, 1, 0\}$. Dotted gates do not affect OUTPUT-gate even if the gate negate output, so effective circuit do not include them.

**Theorem 1.6.**

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

*Proof.* Because all input variable pair is $\{01, 10\}$ and do not become $\{00, 11\}$ in one input, and NNF circuit is almost monotone circuit, so effective circuit have to to join another input effective circuit with OR-gate before connecting OUTPUT-gate.

Figure 1.7 shows example of connectivg effective circuit which previous 1.1 and input $\{x_1, x_2, x_3\} = \{1, 1, 0\}, \{0, 0, 1\}$. Effective circuit include one of input variable

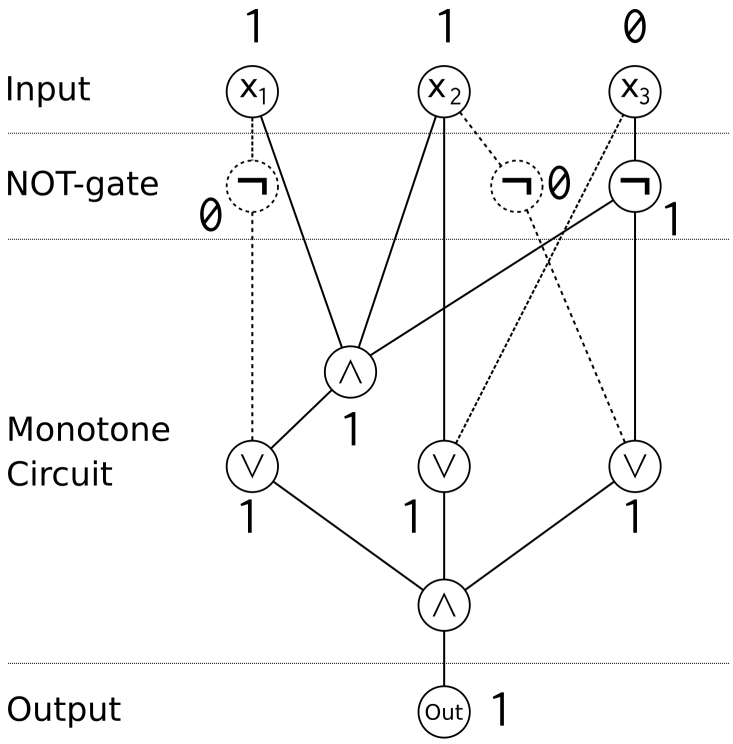


FIGURE 1.6. Effective circuit

pair, and other side of variable pair do not become 1. So AND-gate cannot meet another effective circuit.

□

**Theorem 1.7.**

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

*That is, each different valiables correspond to unique AND-gate.*

*Proof.* Mentioned above 1.6, 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.

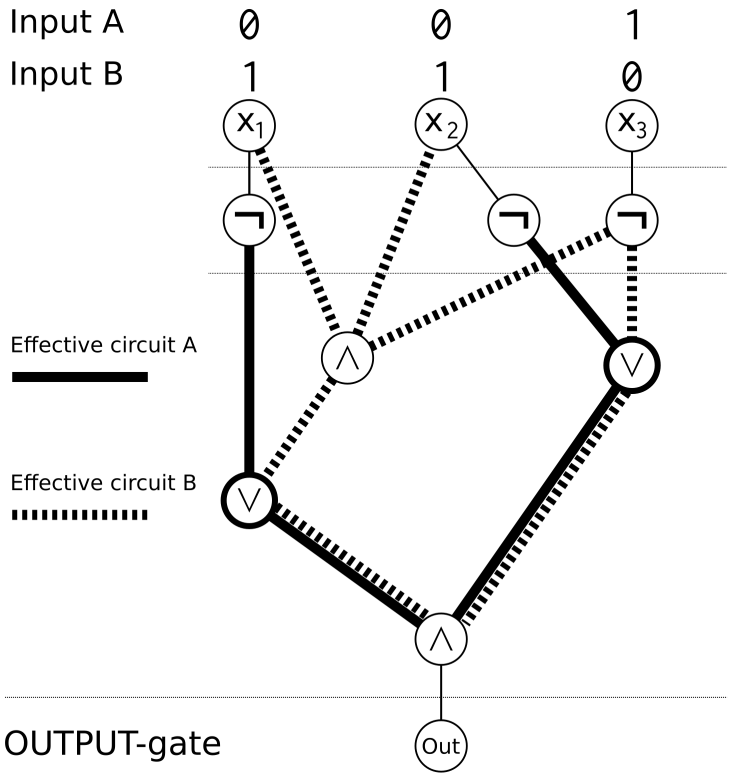


FIGURE 1.7. Different variables independency

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 input AND-gate does not include these different variables.

Case b), because no boundary input become accept input, some OR-gate which join neighbor input 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 1.9. Such AND-gate become 1 if and only if input include different variables of one side of neighbor input pair. That is, AND-gate correspond to each different variables.

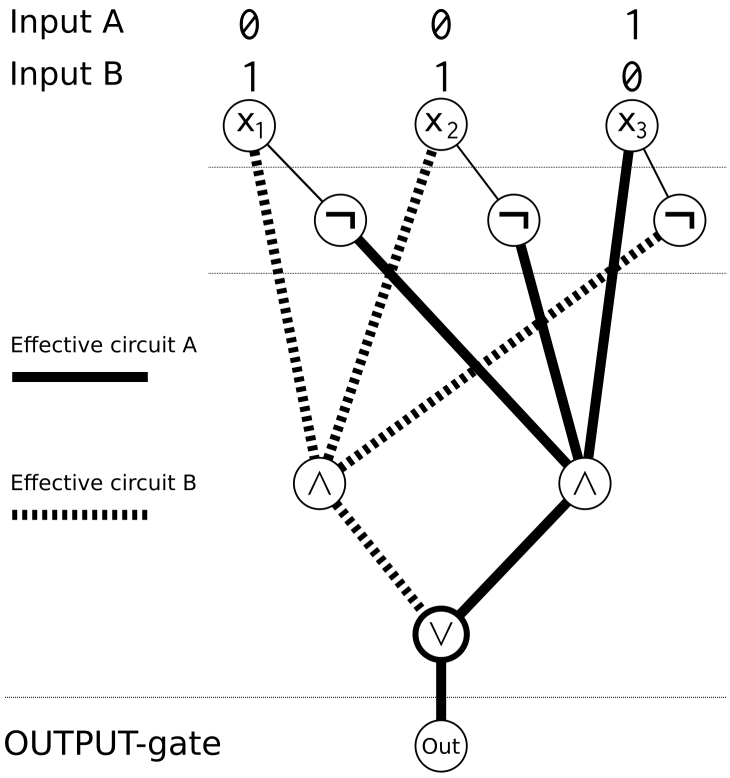


FIGURE 1.8. Example of a)

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

That is to say, we can measure NNF circuit family size (that equal DTM computing time) by counting different variables of problem's sandwich structure.

## 2. Sandwich structure

Consider sandwich structure of some problems. For example, $t_a = (x_1 \vee \overline{x_2}) \wedge (\overline{x_1} \vee \overline{x_3}) \wedge x_2 \wedge x_3 \wedge x_4$ is one of negation of Horn SAT problem $\overline{HornSAT}$ (Accept if and only if Horn CNF is $\bot$). We can reduce $t_a$ easily to another accept input $t_r = (x_1 \_\_\_) \wedge (\overline{x_1} \vee \overline{x_3}) \wedge x_2 \wedge x_3 \wedge x_4$ (_: filler). This means that each $\overline{HornSAT}$ input are close each other, and number of $\overline{HornSAT}$ different variables type is not many.

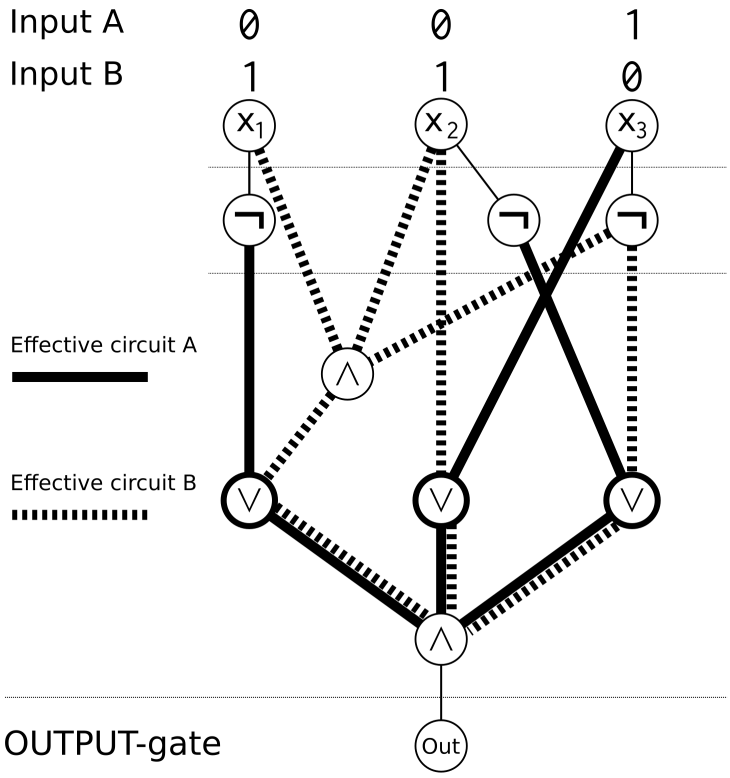


FIGURE 1.9. Example of b)

This is one of reason that we can compute P problem easily.

**Theorem 2.1.**

*Let $\overline{HornSAT}$ (negation of Horn SAT problem) as problem if and only if Horn CNF is $\bot$.*

*In $\overline{HornSAT}$, there is some sandwich structure which number of different variables is at most polynomial size.*

*Proof.* $t \in \overline{HornSAT}$ can reduce another $t' \in \overline{HornSAT}$ that delete all literal $\overline{x_i}$ which $x_i = c \in t$. Therefore, $t$ can reduce $t'$ by deleting $\overline{x_i}$ one by one. From the view of sandwich structure, $\overline{x_i}$ can delete within constant different variables if $\overline{HornSAT}$ have ignore symbol like $\overline{\blacksquare_i}$. Therefore, number of different variables of $t \in \overline{HornSAT}$ is at most constant size.

Consider number of different variables type in $\overline{HornSAT}$. Because number of different variables is at most constant size, number of different variables type is combination of different variables;

$\begin{pmatrix} |t| \\ k \end{pmatrix} = \frac{|t|!}{k! \times (|t|-k)!} < O\left(|t|^k\right)$

k : length of ■ code

Therefore 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