Limiting Negations in Bounded Treewidth and Upward Planar Circuits

Jing He[⋆] Hongyu Liang[⋆] Jayalal Sarma M.N.[*]

Abstract

The decrease of a Boolean function $f : \{0,1\}^n \to \{0,1\}$, denoted by $d(f)$ is the maximum number of inverse indices in any increasing chain of inputs $x_1, \ldots, x_\ell \in \{0,1\}^n$, where i is called an inverse index if $f(x_i) > f(x_{i+1})$. It follows from a theorem of Markov (JACM 1958) that the minimum number of negation gates in a circuit sufficient to compute any Boolean function f is $\lceil \log(d(f)+1) \rceil$ and that this is necessary in general. A recent result due to Morizumi (ICALP 2009) proves that $d(f)$ negations are necessary and sufficient for computing any Boolean function f by a formula. In this paper we explore the situation in between formulas(directed trees) and general circuits(directed acyclic graphs), and related models. We obtain the following results:

1. We argue that for any Boolean function f, there is a circuit computing f, that uses $\lceil \log(d(f)+1) \rceil$ negations and has treewidth at most $\lceil \log(d(f)+1) \rceil + 1$. For $1 \leq k \leq \lceil \log(d(f)+1) \rceil$, we prove that $d(f) \cdot 8k/2^k$ negations are sufficient to compute any Boolean function f by circuits of treewidth at most k. Moreover, if there is a circuit family of size $s = s(n)$ and treewidth $k = k(n)$ computing $\{f_n\}$, then there exists a circuit family of size $s \cdot n^{O(1)} \cdot 2^{O(\min\{k, \log n\})}$ and treewidth at most $2k$ which computes $\{f_n\}$ and contains $O(\max\{nk/2^{2k}, \log n\})$ negation gates.
2. We obtain tight bounds on the number of negation gates required to compute specific functions such as $\textsc{Parity}_n, \overline{\textsc{Parity}_n}$ and $\textsc{Inverter}_n$ by one-input-face upward planar circuits. We extend the lower bounds in this case to a larger class $\mathbb{W}$ of functions which also includes natural functions like $\textsc{Add}$ and $\textsc{Subtract}$. For functions in this class, we show a direct sum theorem by proving a tight lower bound on the number of negations required to compute t functions simultaneously using a one-input-face upward planar circuit.
3. We demonstrate the limitations of the one-input-face constraint in the upward planar circuits by showing the explicit function which can be computed by a monotone upward planar circuit, but cannot be computed by any montone one-input-face upward planar circuit.

[*]Institute for Theoretical Computer Science, Tsinghua University, Beijing, China. Email:`hejing2929@gmail.com, hongyuliang86@gmail.com, jayalal@tsinghua.edu.cn`. This work was supported in part by the National Natural Science Foundation of China Grant 60553001, and the National Basic Research Program of China Grant 2007CB807900,2007CB807901.

4. We prove that for every Boolean function f, there exists a multilective upward planar circuit which uses at most $\lceil \frac{d(f)+1}{2} \rceil$ negation gates for computing f.

1 Introduction

Proving super-polynomial size lower bounds for circuits computing explicit functions in NP is a central problem in circuit complexity theory. Theory of monotonicity in Boolean circuits became fruitful in this context and it culminated in the exponential size lower bound due to Razborov [14] for any monotone circuit computing the clique function. Tardos [18] demonstrated that there are explicit functions in P which requires exponential size for monotone circuits computing them. This area received a lot of attention and several important resource lower bounds were proved against monotone Boolean circuits [7, 13]. Subsequent research went into exploring how far the monotonicity constraint can be relaxed by allowing a small number of negation gates and still obtain exponential(or super-polynomial) size lower bounds for circuits computing an explicit function. An important step in this direction was made by Amano and Maruoka [1], who proved exponential size lower bounds for an explicit function when the circuit is allowed only $\frac{1}{6} \log \log n$ negations.

How far can one improve the above lower bound result in terms of the number of negations allowed in the circuit? This highlights the importance of exploring the power of negation gates in Boolean circuits. A very fundamental question in this direction is about the number of negations that is required to compute any Boolean function, called the inversion complexity of the function. Historically much earlier (in 1958), Markov [9] came up with a surprisingly tight bound for the inversion complexity of any Boolean function. Let $f : \{0,1\}^n \to \{0,1\}$, and let $(x_1, x_2, \ldots, x_n)$ and $(y_1, y_2, \ldots, y_n)$ be two Boolean vectors in $\{0,1\}^n$. Define $x \leq y$ if $x_i \leq y_i$ for all i. The decrease of function f with respect to an increasing chain of Boolean vectors $v_1, \ldots, v_m \in \{0,1\}^n$, is defined as the number of i such that $f(v_i) > f(v_{i+1})$. The decrease of the function, denoted by $d(f)$ is the maximum decrease over all increasing chains of Boolean vectors. Thus, the decrease of the function f can at most be n. Markov [9] showed a tight characterization of the inversion complexity of a function f as $\lceil \log(d(f)+1) \rceil$. This implies that, to compute a Boolean function on n variables, $\lceil \log(n+1) \rceil$ negation gates are sufficient.

However, the circuits that Markov constructed are of exponential size although they use only $O(\log n)$ negations. Complementing this, Fischer [5] showed that for every poly-sized circuit, there is an equivalent poly-sized circuit which uses at most $\lceil \log(n+1) \rceil$ negations. This in particular implies that if Amano and Maruoka's result is improved to handle up to $O(\log n)$ negations it will already *separate* P *from* NP. However, since this goal seems to be elusive, it is natural to look for restricted circuit classes for which we can improve the number of negations (beyond $O(\log \log n)$) in the circuit classes against which the lower bounds are proven. In this program, the first step is to study the computational power of the negations in these limited circuit classes, which motivates this work.

Many years after Markov's and Fischer's results, Santha and Wilson [15] showed a contrasting picture in the constant depth world: there are functions requiring super-logarithmic number of negation gates in any poly-sized constant-depth circuit computing them. This was further extended to bounded depth circuits in [17]. Recently, Morizumi [12] studied the case of formulas and proved tight lower and upper bounds for the inversion complexity of Boolean functions. More precisely, he proved that the inversion complexity in formulas computing the function f is exactly $d(f)$. He also proved an analogue of Fischer's result in this context: if there is a polynomial size formula computing a function f then there is a polynomial size formula for f which uses at most $\lceil \frac{n}{2} \rceil$ number of negations.

In this paper, we study circuit classes between formulas (directed trees) and general circuits (directed acyclic graphs). Many parameters interpolate between the two. Two important ones we consider here are upward planarity and treewidth of the underlying undirected graph of the circuit. We defer the formal definition of these parameters to Section 2.

Roughly speaking, treewidth measures how formula-like the circuit is (treewidth of formulas is 1). We consider circuits of treewidth k, and parameterize the number of negations in the circuit in terms of k. Power of such circuits has been considered earlier in the context of classical [6] and quantum computation [10]. As noted in [6], leveled circuits (or graphs) of width k has treewidth at most $2k-1$. In the other direction, they prove that circuits of $\log^i n$ treewidth can be simulated by circuits of $\log^{i+1} n$ width without incurring much increase in size. This interleaving of the treewidth (as a resource in the ciruit) with width, and the fact that they are natural parameterized classes between formulas and general circuits, motivates further study of bounded treewidth circuits. For the latter point, it was shown in [6] that the circuits of treewidth k and size s can also be simulated by formulas of size roughly s^{k^2}. However, the number of negations used in the construction is large even if the original circuit is monotone.

We explore the power of such circuits in the context of inversion complexity. To begin with, we argue that treewidth beyond $O(\log n)$ does not help in general. More precisely, for any Boolean function, there is a circuit that uses $\lceil \log(d(f)+1) \rceil$ negations and has treewidth at most $\lceil \log(d(f)+1) \rceil + 1$ (Theorem 3.5). We prove the following general result regarding the upper bound on the inversion complexity in terms of treewidth of the circuit.

Theorem 1.1. *Let f be a non-monotone Boolean function and $1 \leq k \leq \lceil \log(d(f)+1) \rceil$. There exists a circuit of treewidth at most k computing f which contains at most $d(f) \cdot 8k/2^k$ negations.*

However, as in the case of Markov's theorem, the size of the circuit in the above theorem could be exponentially large. Hence complementing this, we show that if a Boolean function can be computed by a circuit family of polynomial size and treewidth k, then it can also be computed by a circuit family of polynomial size, treewidth $2k$ which contains at most $O(\max\{nk/2^{2k}, \log n\})$ negations. We prove this by obtaining the following more general result.

Theorem 1.2. *Let $\{f_n\}$ be a family of Boolean functions. If there is a circuit family of size $s = s(n)$ and treewidth $k = k(n)$ computing $\{f_n\}$, then there exists a circuit family of size $s \cdot n^{O(1)} \cdot 2^{O(\min\{k, \log n\})}$ and treewidth at most $2k$ which computes $\{f_n\}$ and contains $O(\max\{nk/2^{2k}, \log n\})$ negation gates.*

Upward planar circuits are circuits whose underlying graph is upward planar[1]. These circuit classes have been considered in the literature in many contexts [11, 3, 2, 8] with the underlying motivation of understanding the interplay between topological constraints and the well-studied computational resources such as depth, size, negations, space, time etc. An important consideration while defining this model is whether an input label can appear many times in the circuit. We consider two models of upward planar circuits which are also considered by the above authors.

First, we require that each input label appears at most once in the circuit. Under this setting, McColl [11] showed that the threshold function Th_k^n where $n \geq 5$ and $3 \leq k \leq n-2$ cannot be computed by any monotone planar circuit, although the function itself is monotone. Later on, Beynon and Buckle [3] gave an algorithm which, takes as input the truth-table of a function f, tells whether f can be computed by a monotone planar circuit. However, in both papers the authors made an assumption that all input vertices are in one face and all edges in the circuit go upwards in the plane. We mainly deal with this type of circuits, which are better known in the literature as *one-input-face upward planar circuits*. For this model, we explore the inversion complexity of specific functions and prove tight upper and lower bounds for the Parity function and the Inverter function. More specifically we prove the following:

Theorem 1.3. *Let $n \geq 2$, $f \in \{\textsc{Parity}_n, \overline{\textsc{Parity}_n}\}$. The inversion complexity of the function f with respect to one-input-face upward planar circuits is precisely $n-1$. The inversion complexity of* $\textsc{Inverter}_n$ *with respect to one-input-face upward planar circuits is precisely n.*

We generalize this argument further to obtain a non-trivial collection of Boolean functions for which similar bounds hold (Theorem 4.7). We skip the precise definition of this class to the technical sections (Definition 4.6). For functions in this class, we show a direct sum theorem by proving a tight lower bound on the number of negations required to compute t functions simultaneously using a one-input-face upward planar circuit (Theorem 4.11). In addition, we exhibit the limitation of the one-input-face constraint by showing an explicit function which can be computed by a monotone upward planar circuit, but cannot be computed by any montone one-input-face upward planar circuit (Theorem 4.10).

Although formulas are planar, the above model does not includes them since formulas allow input labels to be duplicated. A planar circuit model where the inputs can be duplicated is called *multilective planar circuit* (which was introduced in [16]). Formulas are clearly multilective planar circuits. For this more powerful class of circuits we are able to improve the upper bound on the inversion complexity ($I_{M-UP}(f)$) slightly.

Theorem 1.4. *For every Boolean function f, $I_{M-UP}(f) \leq \lceil \frac{d(f)+1}{2} \rceil$.*

The rest of the paper is organized as follows. In Section 2 we introduce the preliminaries. Section 3 describes the results about bounded treewidth circuits. Section 4 talks about inversion complexity in planar circuits, considering the upward planar and multilective restrictions.

[1]See Section 2 for a precise definition.

2 Preliminaries

In this section, we introduce the basic definitions that we need in this paper. Let $\mathbb{B}_{n,m}$ denote the set of Boolean functions $f : \{0,1\}^n \to \{0,1\}^m$. $\mathbb{B}_n$ stands for $\mathbb{B}_{n,1}$. For a function $f = f(x_1, \dots, x_n)$, we say f *essentially depends on* x_i if $f|_{x_i=0} \neq f|_{x_i=1}$, i.e., the two sub-functions of f obtained by replacing x_i with 0 or 1 are different with one another.

A circuit is an acyclic directed graph, in which all vertices of fan-in 0 (*input gates*) are associated with some variable $x \in \{x_1, \dots, x_n\}$ or a constant $c \in \{0,1\}$, and all other nodes are either $\wedge, \vee$ or $\neg$. The *size* of a circuit is the number of gates contained in it, and the *depth* of a circuit is the length of the longest directed path from any input vertex to any output vertex. We refer to a standard text book [20] for more definitions.

A circuit is called *semilective* if for all $x \in \{x_1, \dots, x_n\}$, at most 1 input vertices in the circuit is associated with x. It is called *multilective* otherwise. It is easy to see that there is essentially no difference between two cases in the general circuit model, since the fan-out of any vertex is unbounded. But this does affect the power of circuits in restrictions like planarity and treewidth bounded circuits that we consider. A *formula* is a multilective circuit all vertices of which have fan-out at most 1.

For a Boolean function f, the *inversion complexity* of f, denoted by $I(f)$, is the minimum number of negation gates contained in any circuit computing f. If restricting the circuits to be formulas (then f should be a single-output function), we get the definition of *inversion complexity of f in formulas*, denoted by $I_F(f)$. The inversion complexity of a family of Boolean functions can be similarly defined, as a function of n. In this notation, Markov [9] proved that $I(f) = \lceil \log(d(f) + 1) \rceil$ for every Boolean function f and Morizumi [12] proved that $I_F(f) = d(f)$. In particular this implies an exponential gap between the number of negation gates required in two different models.

A *tree decomposition* of a graph $G = (V, E)$ is given by a tuple $(T, (X_d)_{d \in V[T]})$, where T is a tree, each X_d is a subset of V called a *bag*, satisfying 1) $\bigcup_{d \in V[T]} X_d = V$, 2) For each edge $(u, v) \in E$, there exists a tree node d with $\{u, v\} \subseteq X_d$, and 3) For each vertex $u \in V$, the set of tree nodes $\{d : u \in X_d\}$ forms a connected subtree of T. Equivalently, for any three vertices $t_1, t_2, t_3 \in V[T]$ such that t_2 lies in the path from t_1 to t_3, it holds that $X_{t_1} \cap X_{t_3} \subseteq X_{t_2}$.

The *width* of the tree decomposition is defined as $\max_d |X_d| - 1$. The *treewidth* $tw(G)$ of a graph G is the minimum width of a tree decomposition of G. We consider circuits whose underlying graph is of bounded treewidth.

A *planar circuit* is a circuit in which each input label appears exactly once and the underlying undirected graph can be embedded on the plane without edge crossings. It is further called an *upward planar circuit* if it has some planar embedding in which all edges go upwards (monotonically increasing in the vertical direction), and is called *one-input-face upward planar* if besides upward planarity all input nodes are placed at the lowest level. For every function f, let $I_{OUP}(f)$ denote the minimum number of negation gates required for computing f by any one-input-face upward planar circuits.

3 Bounded Treewidth Circuits

In this section we consider circuits with bounded treewidth, which naturally generalizes formulas. We allow circuits to be multilective, i.e., they may contain duplicated input variables. By a Boolean function we will mean a single-output Boolean function.

We first collect some basic facts about treewidth that we will be using. Some of them are folklore, but we include the proof for completeness. These statements also appear in Section 3 in-line without proof.

Lemma 3.1. *Let $G = (V, E)$ be an undirected graph, and $G' = (V', E')$ be the subgraph of G induced on $V' \subset V$. Then $tw(G) \le tw(G') + |V| - |V'|$.*

Proof. Adding all vertices in $V \setminus V'$ to every bag of an optimal tree decomposition of G' will yield a tree decomposition of G with treewidth at most $tw(G') + |V| - |V'|$. $\square$ $\square$

Corollary 3.2. *Let $G = (V, E)$ be an undirected graph, and $G' = (V \cup \{v\}, E \cup E')$ where every edge in E' has v as one of its endpoints. Then $tw(G') \le tw(G) + 1$. Let G'' be a graph obtained from G by combining some vertices in V together to form a new vertex. Then $tw(G'') \le tw(G) + 1$.*

Proof. The first statement follows from Lemma 3.1 since G is an induced graph of G'. To prove the second statement, just note that combining some vertices together is equivalent to the following process: First delete these vertices (which will not increase the treewidth), then add a new vertex and add an edge between it and every remaining vertex which was the neighbor of at least one of the deleted vertices. $\square$ $\square$

Lemma 3.3. *Let $G_0 = (V_0, E_0)$ and $G_1 = (V_1, E_1)$ be two undirected graphs such that $|V_0 \cap V_1| = 1$. Denote $G = G_0 \cup G_1 = (V_0 \cup V_1, E_0 \cup E_1)$. Then $tw(G) \le \max\{tw(G_0), tw(G_1)\}$.*

Proof. Let v be the only vertex in both G_0 and G_1. Let $\mathscr{T}_i$ be an optimal tree decomposition of G_i, and B_i be one of the bags of $\mathscr{T}_i$ containing v, for $i = 0, 1$. Add en edge between B_0 and B_1 to obtain a new tree $\mathscr{T}$. Then it is easy to verify that $\mathscr{T}$ is a tree decomposition of $G_0 \cup G_1$. $\square$ $\square$

Corollary 3.4. *Let C_0 and C_1 be two (single-output) circuits. Obtain C by replacing an arbitrary input node of C_0 with a copy of C_1 (that is, identify the output node of C_1 as an input of C_0). Then $tw(C) \le \max\{tw(C_0), tw(C_1)\}$.*

Proof. The statement follows directly from Lemma 3.3. $\square$

3.1 Inversion Complexity in Bounded Treewidth Circuits

Recall that Markov's theorem states that for every Boolean function f, the minimum number of negation gates contained in a circuit computing f is precisely $\lceil \log(d(f) + 1) \rceil$; that is, $I(f) = \lceil \log(d(f) + 1) \rceil$. We will first show that, if we only care the number of negation gates, then excessive treewidth compared to $I(f)$ is useless.

Theorem 3.5. *For every Boolean function f, there exists a circuit computing f which contains exactly $\lceil\log(d(f)+1)\rceil$ negation gates and has treewidth at most $\lceil\log(d(f)+1)\rceil+1$.*

We need the following definition of *connectors*. For any two Boolean functions f_0 and f_1 with the same set of input variables, a *connector of f_0 and f_1* is a function $\mu(y, y', x)$ satisfying that $\mu(i, 1-i, x) = f_i(x)$ for both $i = 0, 1$, where x is the input vector of f_0 and f_1. It was proved by Markov that there always exists a connector of f_0 and f_1 containing at most $\max\{I(f_0), I(f_1)\}$ negation gates, which is then used in the construction of negation-limited circuits. In order to prove our results, we need the following lemma.

Lemma 3.6. *Every pair of Boolean functions $f_0(x)$ and $f_1(x)$ has a connector $\mu(y, y', x)$ which can be computed by a circuit containing* $\max\{I(f_0), I(f_1)\}$ *negation gates and having treewidth at most* $1 + \max\{I(f_0), I(f_1)\}$.

Proof. By induction on $m = \max\{I(f_0), I(f_1)\}$. If $m = 0$ we let $\mu(y, y', x) = (y' \wedge f_0) \vee (y \wedge f_1)$, which can be computed by a monotone formula (since every monotone function has a monotone formula computing it) and hence has treewidth 1.

Now suppose $m \geq 1$ and the statement holds for all $m' < m$. Let C_i be a circuit computing f_i which contains exactly $I(f_i)$ negation gates, for $i \in \{0, 1\}$. We replace the first negation gate (under some topological order) of C_i with a new variable z to obtain a new circuit C_i', and let $f_i'(z, x)$ be the function computed by this circuit. Letting $h_i(x)$ be the monotone function computed at the predecessor of the first negation gate in C_i, we have $f_i(x) = f_i'(\overline{h_i(x)}, x)$. Note that $\max\{f_0', f_1'\} = m - 1$ and thus by the induction hypothesis, they have a connector $\mu'(y, y', (z, x))$ which can be computed by a circuit having treewidth at most m and containing $m - 1$ negation gates.

Observe that a connector of f_0 and f_1 can be constructed by replacing the variable z in μ' with a formula F computing the function $g(y, y', x) = \overline{(y' \wedge h_0(x)) \vee (y \wedge h_1(x))}$. In order to limit the number of negation gates newly introduced, we need to combine all occurrences of z and replace them with exactly one copy of F. It is obvious that the new circuit is a connector of f_0 and f_1 and contains exactly m negation gates, and by Corollaries 3.2 and 3.4, its treewidth is at most $m+1$. (Note that we should use independent copies of variables when constructing the formula F.) □ □

Now we are ready to prove Theorem 3.5 The proof proceeds by making crucial observations about Markov's proof and is by induction on $d(f)$. The case where $d(f) = 1$ is obvious. Suppose $d(f) \geq 2$ and the theorem holds for all f' such that $d(f') < d(f)$. Let $S \subseteq \{0, 1\}^n$ be the set of all input vectors x such that every chain Y starting with x satisfies that $d_Y(f) \leq 2^{I(f)-1} - 1$ (recall $I(f) = \lceil\log(d(f) + 1)\rceil$). Then, for every chain Y ending at a vector $x \notin S$, $d_Y(f) \leq 2^{I(f)-1} - 1$ (otherwise we can find a chain with decrease $\geq 2^{I(f)} \geq d(f) + 1$ by concatenating Y and the chain witnessing $x \notin S$). We also have $(x \in S \text{ and } x \leq y) \Rightarrow y \in S$, implying that the characterization function of S, denoted by h_S, is monotone.

Define two functions $f_0(x)$ and $f_1(x)$ as follows:

$$f_0(x) = \begin{cases} 1 & \text{if } x \in S, \\ f(x) & \text{if } x \notin S, \end{cases}$$

and

$$f_1(x) = \begin{cases} f(x) & \text{if } x \in S, \\ 0 & \text{if } x \notin S. \end{cases}$$

Then we have $I(f_0) = \lceil \log(d(f_0) + 1) \rceil \leq I(f) - 1$ and similarly $I(f_1) \leq I(f) - 1$. Furthermore, for any function $\mu(y', y, x)$ that is a connector of $f_0(x)$ and $f_1(x)$, we have $f(x) = \mu(\overline{h_S(x)}, h_S(x), x)$. Now let μ be the connector found by Lemma 3.6, which can be computed by a circuit C_μ containing $I(f) - 1$ negation gates and having treewidth at most $I(f)$. We combine all input vertices assigned with y' into one vertex, and replace it with a tree computing $\overline{h_S}$. Then replace each input vertex assigned with the variable y by a tree computing h_S. By Corollaries 3.2 and 3.4, this increases the treewidth of C_μ by at most 1. The resulting circuit computes f and contains at most $I(f)$ negation gates. □

Morizumi showed that the minimum number of negation gates contained in a formula computing f is exactly $d(f)$; that is, $I_F(f) = d(f)$. Another translation of this result is that $d(f)$ negation gates are necessary and sufficient for any circuit with treewidth 1 computing f. Therefore, it is natural to ask what happens when we restrict the treewidth of circuits computing f.

Let $I_k(f)$ denote the minimum number of negation gates contained in any circuit computing f with treewidth at most k. Theorem 3.5 and Morizumi's result indicate that $I_1(f) = I_F(f) = d(f)$ and $I_{\lceil \log(d(f)+1) \rceil + 1}(f) = I(f) = \lceil \log(d(f) + 1) \rceil$. We will show an upper bound on $I_k(f)$ for any k lying in between.

Theorem 3.7 (Theorem 1.1 restated). *Let f be a non-monotone Boolean function and $1 \leq k \leq \lceil \log(d(f) + 1) \rceil$. Then $I_k(f) < d(f) \cdot 8k/2^k$.*

Proof. We will prove, by induction on $d(f)$, that $I_k(f) \leq d(f) \cdot 8k/2^k - 1$ for all $1 \leq k \leq \lceil \log(d(f) + 1) \rceil$. The statement is obvious for $d(f) = 1$.

Suppose $d(f) \geq 2$ and the theorem holds for all f' such that $d(f') < d(f)$. Let $S \subseteq \{0,1\}^n$ be the set of all input vectors x such that every chain Y starting with x satisfies that $d_Y(f) \leq d(f)/2$. Then, for every chain Y ending at a vector $x \notin S$, $d_Y(f) \leq d(f)/2$ (otherwise we can find a chain with decrease $\geq d(f) + 1$ by concatenating Y and the chain witnessing $x \notin S$). We also have $(x \in S \text{ and } x \leq y) \Rightarrow y \in S$.

Define two functions $f_0(x)$ and $f_1(x)$ exactly as in the previous proof. Then we have $d(f_0) \leq d(f)/2$ and $d(f_1) \leq d(f)/2$, and

$$f(x) = \begin{cases} f_1(x) & \text{if } x \in S, \\ f_0(x) & \text{if } x \notin S. \end{cases}$$

It is easy to see that $f(x) = (h_S(x) \wedge f_1(x)) \vee (\overline{h_S(x)} \wedge f_0(x))$, where $h_S(x)$ is the characteristic function of the set S, which is monotone and hence can be computed by a monotone formula. This provides a formula-like construction for connecting f_0 and f_1, which preserves the treewidth of f_0 and f_1. More specifically, if there exists C_i computing f_i for both $i = 0, 1$ such that C_i has treewidth at most k and contains at most t negation gates, we can obtain a circuit with treewidth at most k which computes f and contains at most $2t+1$ negation gates. Having $t = d(f) \cdot 4k/2^k - 1$ will suffice.

However, we cannot directly use the induction hypothesis since the condition $k \leq \lceil \log(d(f_i)+1) \rceil$ may be violated. But this can be overcome. For $i \in \{0,1\}$, if $k > \lceil \log(d(f_i)+1) \rceil$ for some $i \in \{0,1\}$, we can get directly from Theorem 3.5 that there exists a circuit computing f_i with treewidth at most k which contains exactly $I(f_i) = \lceil \log(d(f_i)+1) \rceil < k$ negation gates. Since $k \leq \lceil \log(d(f)+1) \rceil$, we have $2^k \leq 2(d(f)+1) \leq 4d(f)$, from which it follows that $I(f_i) \leq k-1 \leq d(f) \cdot 4k/2^k - 1$. Therefore, for both $i = 0,1$, there exists a circuit C_i computing f_i which has treewidth $\leq k$ and contains $\leq d(f) \cdot 4k/2^k - 1$ negation gates. This finishes the induction part of the proof, and hence the theorem follows. $\square$ $\square$

3.2 Inversion Complexity under Polynomial Size Constraints

In this subsection we show Theorem 1.2 stated in the introduction.

Theorem 3.8 (Theorem 1.2 restated)**.** *Let $\{f_n\}$ be a family of Boolean functions. If there is a circuit family of size $s = s(n)$ and treewidth $k = k(n)$ computing $\{f_n\}$, then there exists a circuit family of size $s \cdot n^{O(1)} \cdot 2^{O(\min\{k, \log n\})}$ and treewidth at most $2k$ which computes $\{f_n\}$ and contains $O(\max\{nk/2^{2k}, \log n\})$ negation gates.*

The following is an immediate corollary of Theorem 3.8.

Corollary 3.9. *If $\{f_n\}$ can be computed by a circuit family of polynomial size and treewidth k, then it can also be computed by a circuit family of polynomial size, treewidth $2k$ which contains at most $O(\max\{nk/2^{2k}, \log n\})$ negation gates.*

In the rest of this section we prove Theorem 3.8. Let $f \in \mathbb{B}_n$. Borrowing the notation from [12], we say f' is a *pseudo i^{th} slice* of f iff $f'(x) = f(x)$ for all $x = (x_1, \ldots, x_n)$ such that $\sum_{j=1}^{n} x_j = i$. Let C be a circuit computing f with treewidth k and size s. For $i = 0, 1, \ldots, n$, we construct a circuit $C^{(i)}$, which computes a monotone pseudo i^{th} slice of f, by pushing all negations in C down by the De Morgan's law until they are directly connected to input variables, and then replacing $\overline{x_i}$ with $Th_i^{n-1}(x_{-i})$. Here we use x_{-i} to denote $(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$, and Th_m^n is the threshold function which equals 1 iff at least m out of n input variables are 1.

In order to push the negations down, we may need to make an additional copy (and one copy suffices) of some gates since the circuit is not necessarily a formula. The size of $C^{(i)}$ is at most $2s \cdot n^{O(1)}$ since the number of original gates at most doubles and the threshold functions can be computed by $n^{O(1)}$-sized formulas [19]. Moreover, the treewidth of $C^{(i)}$ is at most $2k$. To see this, we first regard each threshold circuit Th_i^{n-1} as a single node, and call this "fake circuit" $C^{(i)^*}$. The treewidth of $C^{(i)^*}$ is at most $2k$, since we can make an additional copy of the tree decomposition of C, and then combine every two corresponding bags into one "big bag" to obtain a tree decomposition of $C^{(i)^*}$. This at most doubles the treewidth. Next we need to replace the "threshold nodes" with threshold circuits computing them. It follows from Corollary 3.4 that the treewidth of $C^{(i)}$ is at most $\max\{tw(C^{(i)^*}), 1\} = tw(C^{(i)^*}) \leq 2k$ if we use formulas to compute the threshold functions. Hence, we obtain the following.

Lemma 3.10. *For a Boolean function $f \in \mathbb{B}_n$ which can be computed by a circuit of size s and treewidth k, and an integer $i \in \{0, 1, \ldots, n\}$, there exists a monotone circuit of size $s \cdot n^{O(1)}$ and treewidth $2k$ which computes a pseudo i^{th} slice of f.*

We aim to make use of these pseudo slices to reconstruct f so that the number of negation gates can be reduced, and hence prove Theorem 3.8. The basic idea is to divide them into groups each containing some consecutive slices of f. Among each group, we try to find a circuit of limited treewidth and limited number of negation gates which plays the role of a "selector", in the sense that it selects which slice to use according to the number of 1's in the inputs. Finally, a formula serving as a "universal selector" is applied in order to choose the correct group, which will not increase the treewidth according to Corollary 3.4. A trivial grouping is to put exactly one slice in each group. This will result in a circuit of the form $\bigvee_{i=0}^{n} \left(Th_i^n \wedge \overline{Th_{i+1}^n} \wedge C^{(i)} \right)$, which contains as many as n negation gates. Morizumi showed that by gathering two neighbor slices into one group, the number of negation gates used can be reduced to $\lceil n/2 \rceil$. We will prove a more generalized result by collecting 2^{2k} slices in one group. Before presenting the proof, we need to introduce some notations.

We generalize the notion of circuits. Let $X = \{x_1, x_2, \ldots, x_n\}$ be a set of variables, and $\mathcal{H} = \{h_1, h_2 \ldots, h_m\}$ be a set of Boolean functions each taking $x_1, \ldots, x_n$ as inputs. A circuit *augmented with* $\mathcal{H}$ is similarly defined as a normal circuit, except that every fan-in 0 vertex is now assigned with some function $h_i \in \mathcal{H}$. Functions in $\mathcal{H}$ are called *help functions*, and circuits augmented with $\mathcal{H}$ are also called *$\mathcal{H}$-circuits*. The normal definition of circuits can be seen as a special case in which $\mathcal{H} = \{x_1, x_2, \ldots, x_n\}$; that is, the help functions are exactly the variables themselves.

In the following we fix $\mathcal{H} = \{Th_i^n : i = 0, 1, \ldots, n\} \cup \{f^{(i)} : i = 0, 1, \ldots, n\}$, where $f^{(i)}$ is a pseudo i^{th} slice function of f. For every pair of integers a, b such that $0 \leq a \leq b \leq n$, a Boolean function g is called a *(a, b)-selector of f* iff g is a pseudo i^{th} slice function of f for every i such that $a \leq i \leq b$. It follows that $f^{(i)}$ is a (i, i)-selector of f, and f is a $(0, n)$-selector of itself.

For a $\mathcal{H}$-circuit C, we say C is a *(a, b)-selector circuit of f* if it computes a (a, b)-selector of f. We call it a *good (a, b)-selector circuit of f* if in addition it satisfies the following "replacement rule":

Replacement Rule. For every integer r such that $-a \leq r \leq n - b$, if we replace every input vertex Th_i^n of C by Th_{i+r}^n, and replace every input vertex $f^{(i)}$ of C by $f^{(i+r)}$, for every $0 \leq i \leq n$, then the resulting circuit is a $(a + r, b + r)$-selector circuit of f. (See Figure 5 in Appendix A).

The intuition of this type of "good selector" is that applying simple "shifting" to the inputs will suffice to "shift" the selector in terms of parameters, which provides a way to construct selectors recursively from other selectors of smaller size while not increasing the treewidth and number of negation gates used by too much. To be precise, we show the following existence lemma.

Lemma 3.11. *For every $a, b \in \{0, 1, \ldots, n\}$ such that $b - a + 1 = 2^k$ for some integer $k \geq 1$, there is a good (a, b)-selector circuit of f which has size at most 5^k, treewidth at most k and contains at most k negation gates.*

Proof. By induction on k. When $k = 1$, let C be $\left(Th^n_b \wedge f^{(b)}\right) \vee \left(\overline{Th^n_b} \wedge f^{(a)}\right)$. It is easy to verify that C is a good (a, b)-selector circuit of f with size 4, treewidth 1 and contains 1 negation gate.

Now suppose $k \geq 2$ and the theorem holds for all smaller k. Let C_{k-1} be a good $(a, a + 2^{k-1} - 1)$-selector circuit of f which has size at most 5^{k-1}, treewidth at most $k - 1$ and contains at most $k - 1$ negation gates. Let v be an arbitrary input vertex of C_{k-1}. If v is Th^n_i for some i, we replace it with $\left(Th^n_{a+2^{k-1}} \wedge Th^n_{i+2^{k-1}}\right) \vee \left(\overline{Th^n_{a+2^{k-1}}} \wedge Th^n_i\right)$. If v is $f^{(i)}$ for some i, we replace it with $\left(Th^n_{a+2^{k-1}} \wedge f^{(i+2^{k-1})}\right) \vee \left(\overline{Th^n_{a+2^{k-1}}} \wedge f^{(i)}\right)$. After we finish the replacement for every input node v, we combine all the negations which are connected to nodes assigned with $Th^n_{a+2^{k-1}}$ together to form a new node; the reason for doing this is to reuse it multiple times in order to reduce the number of negation gates used. Call the new circuit C_k. (See Figure 5 in Appendix A for an example.) Since only one copy of $\overline{Th^n_{a+2^{k-1}}}$ is used, C_k contains exactly 1 more negation gates than C_{k-1}. The treewidth of C_k is at most k, as we first replace each input vertex with a tree (which will preserve the treewidth of C_{k-1} due to Corollary 3.4) and then combine some vertices together to form a new one (which will increase the treewidth by at most 1 due to Corollary 3.2). To bound the size of C_k, we note that each input node of C_{k-1} is replaced with a circuit of size 4, and the number of input nodes does not exceed the number of gates in C_{k-1}. Therefore, the size of C_k is at most $5^{k-1} + 4 \cdot 5^{k-1} = 5^k$.

It remains to show that C_k is a good (a, b)-selector circuit of f. Due to our construction, when $\sum_{j=1}^n x_j < a + 2^{k-1}$ (that is, $Th^n_{a+2^{k-1}}(x) = 0$), C_k is equivalent to C_{k-1}, and hence C_k is a $(a, a + 2^{k-1} - 1)$-selector circuit of f. When $\sum_{j=1}^n x_j \geq a + 2^{k-1}$, C_k becomes the circuit obtained from C_{k-1} by replacing Th^n_i with $Th^n_{i+2^{k-1}}$ and replacing $f^{(i)}$ with $f^{(i+2^{k-1})}$, which is of course a $(a+2^{k-1}, a+2^k-1)$-selector circuit of f because C_{k-1} is good by the induction hypothesis. Hence, C_k is a (a, b)-selector circuit of f (remember that $b = a + 2^k - 1$). To see why C_k is good, we shift all the parameters (except n) by r for every $-a \leq r \leq b$, and can similarly prove that the resulting circuit is a $(a + r, b + r)$-selector circuit of f. This completes the induction step. □ □

of Theorem 3.8. Let $f \in \mathbb{B}_n$ and C be a circuit computing it with size s and treewidth k. Let t be the minimum integer such that $n \leq 2^t - 1$, and let $n' = 2^t - 1$. Let $k' = \min\{k, t/2\}$. It is clear that $n' \leq 2n - 1$ and $\log n \leq t \leq \lceil \log n \rceil + 1$. We add $n' - n$ dummy input variables to f and regard it as a function in $\mathbb{B}_{n'}$. For $l = 0, 1, \ldots, 2^{t-2k'} - 1$, let C'_l be a $(2^{2k'}l, 2^{2k'}(l + 1) - 1)$-selector circuit of f constructed by Lemma 3.11, which has size at most $5^{2k'}$, treewidth at most $2k'$ and contains at most $2k'$ negations. Let C' be a formula-like circuit of the form $\bigvee_{l=0}^{2^{t-2k'}} \left(Th^{n'}_{2^{2k'}l} \wedge \overline{Th^{n'}_{2^{2k'}(l+1)}} \wedge C'_l\right)$ (different terms will use different copies of input vertices). It is easy to see that C computes exactly f and contains at most $(2k' + 1)2^{t-2k'} = (2k' + 1)(n' + 1)/2^{2k'} \leq 2n(2k' + 1)/2^{2k'}$ negation gates. Since $k' = \min\{k, t/2\}$ and $\log n \leq t \leq \lceil \log n \rceil + 1$, this is at most $\max\{2n(2k + 1)/2^{2k}, 2n(t + 1)/2^t\} = \max\{O(nk/2^{2k}), O(\log n)\} = O(\max\{nk/2^{2k}, \log n\})$. Furthermore, C' has size at most $2^{t-2k'}(4 + 5^{2k'}) = n \cdot 2^{O(k')} = n \cdot 2^{O(\min\{k,t/2\})} \leq n \cdot 2^{O(\min\{k,\log n\})}$, and has treewidth at most $2k'$. Note that C' is not a "true" circuit, but one augmented with help functions $\{Th_i^n : i = 0, 1, \ldots, n\} \cup \{f^{(i)} : i = 0, 1, \ldots, n\}$. We replace every input node of C' with a circuit computing the corresponding help function. Since Th_i^n is computable by a poly-sized formula, and by Lemma 3.10 $f^{(i)}$ is computable by a monotone circuit of treewidth $2k$ and size $s \cdot n^{O(1)}$, the resulting circuit has size $s \cdot n^{O(1)} \cdot 2^{O(\min\{k, \log n\})}$ and treewidth at most $\max\{2k, 2k'\} = 2k$ due to Corollary 3.4. This finishes the proof of Theorem 3.8. $\square$ $\square$

4 Inversion Complexity in Planar Circuits

4.1 Lower Bounds for One-Input-Face Upward Planar Circuits

In this section we will focus on the inversion complexity in one-input-face upward planar circuits. We will prove $\Omega(n)$ lower bounds of $I_{OUP}(f)$ for many functions f, including some tight results. Since $I(f) = O(\log n)$ for all $f \in \mathbb{B}_{n,m}$, an exponential gap between the number of negation gates used in general circuits and one-input-face upward planar circuits is obtained for a number of natural functions, including Parity, Inverter, Add and Subtract.

Theorem 4.1. *For $f \in \{\text{Parity}_n, \overline{\text{Parity}_n}\}$ where $n \geq 2$, $I_{OUP}(f) = n - 1$.*

Proof. To prove that $n-1$ is an upper bound, just notice that $\overline{\text{Parity}_n} = \text{Equiv}(\text{Parity}_{n-1}(x_1, \ldots, x_{n-1}),$ ($\text{Equiv}(x, y)$ computes $x \equiv y$), and that both the XOR gate and the Equiv gate can be simulated by planar circuits each containing one negation gate (see Figure 3 for the standard construction).

Next we prove that $I_{OUP}(f) \geq n - 1$ for $f \in \{\text{Parity}_n, \overline{\text{Parity}_n}\}$ where $n \geq 2$. The statement will be proved by induction on n. It is obvious when $n = 2$, since both functions are non-monotone. Now suppose $n \geq 3$ and the statement holds for all $n' < n$. Let $f \in \{\text{Parity}_n, \overline{\text{Parity}_n}\}$. Assume that the theorem doesn't hold for f, and let C be a one-input-face upward planar circuit computing f which contains at most $n - 2$ negation gates. Without loss of generality, we may assume that the order of the input variables are $x_1, \ldots, x_n$ if we read them from left to right. (The following argument holds for all input orders.)

Suppose G is the predecessor of the first negation gate in C (under some topological order of the underlying graph of C). Let g denote the function computed at the gate G.

Let $S = \{x_i \mid g \text{ essentially depends on } x_i, 1 \leq i \leq n\}$. Let $l = \min\{i \mid x_i \in S\}$ and $r = \max\{i \mid x_i \in S\}$. Denote by PI the set of all prime implicants of g. Since g is monotone, every prime implicant of it only contains positive literals. We first prove the following lemmas.

Lemma 4.2. *$S = \{x_i \mid l \leq i \leq r\}$.*

Proof. Assume the contrary that $x_i \notin S$ for some $l < i < r$. We set all variables in S to 1. Since g is monotone, this will fix G to be 1 no matter what values the other variables are assigned with. Hence we can find one path from x_l to G on which each gate is fixed to be 1, and another path from x_r to G with the same property. Due to the special property of

Parity, f essentially depends on x_i even after all variables in S are set to 1. Since C is one-input-face upward planar and f is non-monotone, the output gate of C (denoted by O) must lie out of the area α. Again using the upward planarity of C, we know that every path connecting x_i and O must intersect the two 1-paths previously found. Thus the variable x_i is "disconnected" from O, in the sense that changing the value of x_i will not affect the value of O (since all gates on the 1-paths are fixed to be 1), a contradiction. (See Figure 1 for an illustration.) □ □

Lemma 4.3. *Let $p = x_{i_1}x_{i_2}\dots x_{i_k} \in PI$ be any prime implicant of g, where $l \le i_1 < \dots < i_k \le r$. Then $i_m = i_{m-1} + 1$ for all $2 \le m \le k$; that is, any prime implicant is in fact an "interval".*

Proof. The proof is similar to that of Lemma 4.2. We set all variables in p to 1, which fixes G to evaluate to 1. Then any variable between x_{i_1} and x_{i_k} which doesn't belong to p (if any) will be disconnected from the output gate of C, which is a contradiction. □ □

Lemma 4.4. *Each prime implicant of g has size at least 2.*

Proof. Assume the contrary that $x_i \in PI$ for some i. Setting $x_i = 1$ will fix G to evaluate to 1, and thus at least one negation gate can be eliminated from C. The resulting circuit will compute Parity_{n-1} or $\overline{\text{Parity}_{n-1}}$, and contains at most $n-3$ negation gates. This contracts our induction hypothesis. □ □

Lemma 4.5. *Let p_l and p_r be the prime implicants of g containing x_l and x_r, respectively. Then p_l and p_r do not intersect with each other; that is, they contain no common variables.*

Proof. Suppose p_l and p_r both contain some variable, say, x_i. From Lemmas 4.2 and 4.3 we know that each prime implicant of g must contain x_i as well. Thus we can set $x_i = 0$ which will fix G to be 0, and eliminate at least one negation gate from C. The rest of the proof is similar to that of Lemma 4.4. □ □

Note that Lemma 4.5 implies that PI contains at least 2 different prime implicants. Now we are ready to prove Theorem 4.1. Let C_G be the induced sub-circuit of C with output gate G. More precisely, C_G contains all vertices of C (variables and gates) from which G is reachable, and all edges spanning them. So C_G computes the function g. Let $p_l = x_1x_2\dots x_j$ and $p_r = x_kx_{k+1}\dots x_r$ be the prime implicants of g containing x_l and x_r, respectively. By Lemmas 4.4 and 4.5 we have $l < j < k < r$. Imagine the scenario where all variables except x_k are set to 0. Since f still essentially depends on x_k after this restriction, there exists a "switching path" $\mathcal{P}$ from x_k to the output gate of C such that flipping the value of x_k will cause all gates on $\mathcal{P}$ to change their values (given that other variables are set to 0). Also note that the output gate is not contained within the area of C_G. Due to the upward planarity of C, $\mathcal{P}$ must intersect the boundary of C_G. Let $\mathcal{P}_{in}$ denote the inside part of $\mathcal{P}$ respect to C_G. As C_G is monotone, any gate on $\mathcal{P}_{in}$ switches from 0 to 1 if x_k switches from 0 to 1, given that other variables are 0. Therefore, setting x_k to 1 will fix all gates on $\mathcal{P}_{in}$ to evaluate to 1, regardless of the assignment to other variables (use the monotonicity again). By Lemma 4.4, G doesn't lie on $\mathcal{P}_{in}$. There are two cases to examine.

Case 1: $\mathcal{P}_{in}$ intersects the right boundary of C_G. We set $x_l = x_{l+1} = \ldots = x_{k-1} = 0$ and $x_k = x_{k+1} = \ldots = x_{r-1} = 1$. Under this restriction G will compute exactly x_r. But x_r cannot affect G now, since any path from x_r to G must intersect $\mathcal{P}_{in}$ (see Figure 2). This leads to a contradiction.

Case 2: $\mathcal{P}_{in}$ intersects the left boundary of C_G. We set $x_1 = x_2 = \ldots = x_{j-1} = 1$, $x_{j+1} = \ldots = x_{k-1} = 0$, $x_k = 1$ and $x_{k+1} = \ldots = x_r = 0$. Under this restriction G will compute exactly x_j. But any path from x_j to G must intersect $\mathcal{P}_{in}$, which again gives a contradiction (see Figure 2). We have finished the induction part, and hence the theorem follows. □

We next introduce the classes of functions for which we can apply a similar argument and prove linear lower bounds on $I_{OUP}(f)$. Although the definition seems restrictive, it will be shown that many natural functions fall into these classes.

Definition 4.6. *Let $k, n_0, n, m \in \mathbb{N}, k \geq 1$ and $f \in \mathbb{B}_{n,m}$ be a function with input set X and output set Y. f belongs to the class $\mathbb{W}^n_{k,n_0}$ if and only if it satisfies:*

1. *If $n > n_0$, then for any variable $x \in X$ and any "uniform" restriction σ which maps all variables in $X \setminus \{x\}$ to the same Boolean constant $b \in \{0, 1\}$, there exists a non-monotone output $y \in Y$ such that $y|_\sigma$ essentially depends on x.*

2. *If $n > n_0$, then for any variable $x_0 \in X$ and any constant $c_0 \in \{0, 1\}$, there exists a set of k' variables $\{x_1, \ldots, x_{k'}\} \subseteq X \setminus \{x_0\}$ and a sequence of Boolean constants $c_1, \ldots, c_{k'}$, where $0 \leq k' \leq \min\{k-1, n-1\}$, such that $f|_\gamma \in \mathbb{W}^{n-k'-1}_{k,n_0}$, where γ denotes the restriction which maps x_i to c_i for all $0 \leq i \leq k'$.*

We explain the intuition a little. The first property in the above definition requires the n-argument function to be nontrivial even after $n-1$ variables of it are fixed (to the same value). Also, the function itself should be non-monotone because this is important in the following proof. Note that we only require the original function to be non-monotone, but not the sub-function under the restriction. The second property characterizes a feature of "self-reducibility", indicating that induction may be useful for proving properties of these functions. The condition $n > n_0$ is used to avoid degenerated cases. By the definition, we have $\bigcup_{m\in\mathbb{N}} \mathbb{B}_{n,m} \subseteq \mathbb{W}^n_{k,n_0}$ for all $n \leq n_0$, and $\mathbb{W}^n_{k,n_0} \subseteq \mathbb{W}^n_{k,n_0'}$ for any $n_0 \leq n_0'$.

Now we state the theorem about the functions in this class.

Theorem 4.7. *$I_{OUP}(f) \geq \lceil \frac{n-n_0}{k} \rceil$ if $f \in \mathbb{W}^n_{k,n_0}$.*

Proof. The proof of Theorem 4.7 is very similar to the previous proof and the necessary modifications are described below: We use induction on n. The base cases $n \leq n_0$ are trivial. Now suppose $n > n_0$, and the theorem holds for all functions in $\mathbb{W}^m_{k,n_0}$ with $m < n$, but doesn't hold for some $f \in \mathbb{W}^n_{k,n_0}$. Let C be a one-input-face upward planar circuit computing f which contains strictly less than $\lceil \frac{n-n_0}{k} \rceil$ negation gates. We reserve all notations used in the proof of Theorem 4.1. Then, Lemmas 4.2,4.3,4.4 and 4.5 still hold. We briefly show the modified argument for them.

For Lemma 4.2, previously we argued that if some x_i is not in S, then it will be disconnected from the output gate of C, contradicting the property of $\textsc{Parity}$. Now we have $f \in \mathbb{W}^n_{k,n_0}$, so by the first condition of its definition, there exists a non-monotone output gate H whose corresponding function h essentially depends on x_i even after setting all variables in S to 1. Since G is before the first negation gate in C, H must lie out of the area α, and will not be affected by x_i due to the two 1-paths, a contradiction. Lemma 4.3 is similar.

For Lemma 4.4, suppose setting $x_i = 1$ will fix G to 1. Note that by the second condition of Definition 4.6, we can find another $k' \leq k-1$ variables such that fixing them to be some constants will make f degenerate to some function $f' \in \mathbb{W}^{n-k'-1}_{k,n_0}$. The resulting circuit computing f' has strictly less than $\lceil\frac{n-n_0}{k}\rceil - 1 = \lceil\frac{n-k-n_0}{k}\rceil$ negation gates, since we eliminated at least one negation gate from C. This leads to a contradiction since by induction hypothesis, $I_{OUP}(f') \geq \lceil\frac{n-k'-1-n_0}{k}\rceil \geq \lceil\frac{n-k-n_0}{k}\rceil$. Lemma 4.5 is similar.

In the rest part of the original proof, we only use the property of $\textsc{Parity}$ once: The output essentially depends on x_k even if all variables other than x_k are set to 0. Now we replace $\textsc{Parity}$ with $f \in \mathbb{W}^n_{k,n_0}$. By the first condition of Definition 4.6, we can find a non-monotone output of f with the same desired property, and the proof goes through analogously. □ □

Corollary 4.8. *The following statements hold:*
(1) $I_{OUP}(\textsc{Inverter}_n) = n$ for all $n \geq 1$.
(2) $I_{OUP}(\textsc{OrOfParity}_{n_1,\ldots,n_t}) \geq \frac{\sum_{i=1}^t n_i}{2}$ for all nonnegative even numbers $n_1, n_2, \ldots, n_t$.
(3) For $f \in \{\textsc{Add}_{2n}, \textsc{Subtract}_{2n}\}$ where $n \geq 1$, $I_{OUP}(f) \geq n$.

Proof. We need to know which functions are included in this definition. The function $\textsc{Parity}_n$ is defined as follows: $\textsc{Parity}_n(x_1, x_2, \ldots, x_n) = 1$ iff $|\{i : x_i = 1, 1 \leq i \leq n\}|$ is odd. It can be verified (by simple induction) that both $\textsc{Parity}_n$ and $\overline{\textsc{Parity}_n}$ belong to $\mathbb{W}^n_{1,1}$, for all $n \geq 1$. By the observation that $\textsc{Parity}_1$ is monotone and thus not contained in $\mathbb{W}^1_{1,0}$, we can inductively prove that $\textsc{Parity}_n \notin \mathbb{W}^n_{1,0}$ and $\overline{\textsc{Parity}_n} \notin \mathbb{W}^n_{1,0}$ for any $n \geq 1$.

Given t nonnegative even numbers $n_1, n_2, \ldots, n_t$, we define the following function $\textsc{OrOfParity}_{n_1,\ldots,n_t} \in \mathbb{B}_{\sum_{i=1}^t n_i}$:

$$\textsc{OrOfParity}_{n_1,\ldots,n_t}(x) = \bigvee_{i=1}^{t} \textsc{Parity}_{n_i}(x_{i,1}, x_{i,2}, \ldots, x_{i,n_i}).$$

Given such a function, imagine that all variables but $x_{i,j}$ are set to 0 (or 1). Since all n_i's are even, the resulting function computes $x_{i,j}$ (or $\overline{x_{i,j}}$) and thus essentially depends on it. If we fix some $x_{i,j}$ to a constant b, then fixing another variable $x_{i,j'}$ with $j' \neq j$ will make the function become $\textsc{OrOfParity}_{n_1,\ldots,n_{i-1},n_i-2,n_{i+1},\ldots,n_t}$. Due to this recursive property, one can prove that $\textsc{OrOfParity}_{n_1,\ldots,n_t} \in \mathbb{W}^{\sum_{i=1}^t n_i}_{2,0}$ for all nonnegative even numbers $n_1, n_2, \ldots, n_t$.

We next come to multi-output functions. Define the function $\textsc{Inverter}$ as $\textsc{Inverter}_n(x_1, x_2, \ldots, x_n) = (\overline{x_1}, \overline{x_2}, \ldots, \overline{x_n})$. We can easily prove $\textsc{Inverter}_n \in \mathbb{W}^n_{1,0}$ inductively, for all $n \geq 1$.

We also have $\textsc{Add}_{2n} \in \mathbb{W}^{2n}_{2,0}$ for all $n \geq 1$, where $\textsc{Add}_{2n}(x_1, x_2, \ldots, x_{2n})$ computes the sum of two binary numbers $\overline{x_1x_2\ldots x_n}$ and $\overline{x_{n+1}x_{n+2}\ldots x_{2n}}$. To see this, first note that every output of $\textsc{Add}_{2n}$ is non-monotone, and at least one of the outputs essentially depends on x_i even if all variables other than x_i are arbitrarily fixed. To validate the second property, we point out that after fixing x_i to c_i for some $i \in \{1, \ldots, n\}$ (resp. $i \in \{n+1, \ldots, 2n\}$) and $c_i \in \{0, 1\}$, a further restriction which fixes x_{i+n} (resp. x_{i-n}) to $1 - c_i$ will reduce the function to $\textsc{Add}_{2n-2}$. By a similar argument, we can also prove that $\textsc{Subtract}_{2n} \in \mathbb{W}^{2n}_{2,0}$, where $\textsc{Subtract}$ computes the difference between two binary numbers and can be defined easily.

Now the lower bounds are direct from Theorem 4.7, and the the upper bound for $\textsc{Inverter}$ is due to the trivial construction. □ □

A Direct Sum Theorem: We consider the direct sum of Boolean functions, i.e., a collection of different Boolean functions on pairwise disjoint variable sets. More precisely, given a set of Boolean functions $\{f_1, f_2, \ldots, f_t\}$, where $f_i \in \mathbb{B}_{n_i,m_i}$, the *direct sum of* $f_1, f_2, \ldots, f_t$ is a Boolean function $f \in \mathbb{B}_{\sum_{i=1}^t n_i, \sum_{i=1}^t m_i}$ with a set of input variables $\{x_{i,j} : 1 \leq i \leq t, 1 \leq j \leq n_i\}$, which is defined as

$$\begin{aligned} & f(x_{1,1}, \ldots, x_{1,n_1}, x_{2,1}, \ldots, x_{2,n_2}, \ldots, x_{t,1}, \ldots, x_{t,n_t}) \\ = \; & (f_1(x_{1,1}, \ldots, x_{1,n_1}), f_2(x_{2,1}, \ldots, x_{2,n_2}), \ldots, f_t(x_{t,1}, \ldots, x_{t,n_t})). \end{aligned}$$

If f is the direct sum of $f_1, \ldots, f_t$, then each of the functions is called an *element* of f. What is the relationship between the inversion complexity of f and that of its elements? Trivially $I(f) \leq \sum_{i=1}^t I(f_i)$, but the result is far from tight. To see this, let $f_i \in \mathbb{B}_n$ be a function of decrease $\Theta(n)$ (e.g. $\textsc{Parity}$) for all $i \in \{1, 2, \ldots, t\}$. From Markov's theorem we have $I(f_i) = \lceil \log(d(f_i) + 1) \rceil = \Theta(\log n)$, and $I(f) \leq \lceil \log(tn + 1) \rceil$. Taking $t = n^{\Theta(1)}$ gives $I(f) \leq O(\log n)$ and $\sum_{i=1}^t I(f_i) = n^{\Theta(1)}$, where a large gap between $I(f)$ and $\sum_{i=1}^t I(f_i)$ appears. This indicates that computing the direct sum of functions (with large decreases) can benefit from interconnections between its seemingly independent elements. We will show that this is not always the case if we adopt planar circuits as our computation model.

Theorem 4.9. *Let $f_i \in \mathbb{W}^{n_i}_{k_i,m_i}$ for all $i \in \{1, \ldots, t\}$, if f is the direct sum of f_is, then $I_{OUP}(f) \geq \sum_{i=1}^t \lceil \frac{n_i - m_i}{k_i} \rceil$.*

of Theorem 4.9. It is similar to the proof of Theorem 4.7. For a collection of functions $f_1, f_2, \ldots, f_t$ where $f_i \in \mathbb{W}^{n_i}_{k_i,m_i}$, and their direct sum f, we call the vector $(t, n_1, n_2, \ldots, n_t)$ the *representative vector* of f. Given two representative vectors $v_i = (t_i, n_{i,1}, n_{i,2}, \ldots, n_{i,t_i})$, $i = 0, 1$, we say $v_0 < v_1$ if:
(1) $t_0 < t_1$, or
(2) $t_0 = t_1$ and $\exists j \in \{1, 2, \ldots, t_0\}$ such that $n_{0,j'} = n_{1,j'}$ for all $1 \leq j' < j$ and $n_{0,j} < n_{1,j}$.

This defines a total order on all representative vectors v, and our proof will be by induction on v. The base case $t = 1$ is exactly Theorem 4.7. For the induction step, we can similarly prove all the four lemmas and finish the rest part of the proof. The only difference is that, when dealing with a special input variable x_i, we need to identify which function it belongs to.

Take the proof of Lemma 4.4 as an illustration. Suppose PI contains a prime implicant of size 1, say x_{i,j_i}. We assume $n_i > m_i$, since otherwise we can just ignore f_i and apply the induction hypothesis. We set x_{i,j_i} to 1. By the second condition of Definition 4.6, we can fix another $k' \leq k_i - 1$ variables from $\{x_{i,j} : 1 \leq j \leq n_i\}$ to make f_i degenerate to some $f'_i \in \mathbb{W}^{n_i-k'-1}_{k_i,m_i}$. The new direct sum function $f' = (f_1, \ldots, f_{i-1}, f'_i, f_{i+1}, \ldots, f_t)$ has a representative vector less than f, so we can use the induction hypothesis to prove the lemma. □ □

It is straightforward from Theorems 4.1,4.9 and Corollary 4.8 that $I(f) = \sum_{i=1}^{t} I(f_i)$ if each f_i is $\textsc{Parity}$, $\overline{\textsc{Parity}}$ or $\textsc{Inverter}$, and f is the direct sum of all f_i's. This differs from the situation in general circuits.

Limitation of the One-Input-Face Constraint: We show that restricting all input vertices to be on the same face (or equivalently on the exterior face, or at the lowest level in the plane) may increase the number of negation gates used for computing some functions. We prove a stronger result, by showing a monotone (multi-output) function which has a monotone upward planar circuit computing it, but cannot be computed by any monotone one-input-face cylindrical circuits (and hence not computable by any monotone one-input-face upward planar circuits). Here a circuit is called *cylindrical* if it can be embedded on a cylinder surface without edge crossings and every edge goes upwards. It is easy to see that cylindricality generalizes upward planarity.

Let $\textsc{MinMax}_n(x_1, x_2, \ldots, x_n) = (\bigwedge_{i=1}^{n} x_i, \bigvee_{i=1}^{n} x_i)$. Thus $\textsc{MinMax}_n \in \mathbb{B}_{n,2}$ and computes the minimum and maximum values among all input variables. It is easy to construct a monotone upward planar circuit for it (Figure 4 Appendix A).

Theorem 4.10. *Let $n \geq 3$. Then $\textsc{MinMax}_n$ can be computed by a monotone upward planar circuit, but cannot be computed by any monotone one-input-face cylindrical circuit.*

Proof. The upward planar circuit computing $\textsc{MinMax}_n$ can be constructed easily, so we only consider the latter inequality. For $n \geq 3$, assume that there is a monotone one-input-face cylindrical circuit C computing $\textsc{MinMax}_n$. According to the equivalent structure of cylindrical circuits, we can regard it as a "disk" in the plane in which all input nodes are on the fringe and all edges go inwards. Let O_1 and O_2 be the output gates computing $\bigvee_{i=1}^{n} x_i$ and $\bigwedge_{i=1}^{n} x_i$ respectively.

For each $i \in \{1, 2, 3\}$, consider the scenario where all variables but x_i is set to 0. Since O_1 computes exactly x_i in this case, we can find a "switching" path $\mathscr{P}_i$ from x_i to O_1, all gates on which evaluate to x_i. Due to the monotonicity of C, setting x_i to 1 will fix all gates on $\mathscr{P}_i$ to 1 no matter what values other variables are taken. The three paths $\mathscr{P}_1$, $\mathscr{P}_2$ and $\mathscr{P}_3$ together with the outer circle containing input variables divide the whole disk into three closed regions, and O_2 must lie inside one of them. (It cannot lie on the boundary, since it computes the $\textsc{And}$ function.) Without loss of generality assume it is contained in the region bounded by $\mathscr{P}_1$ and $\mathscr{P}_2$. Now we set all variables except x_3 to 1, which makes all gates on paths $\mathscr{P}_1$ and $\mathscr{P}_2$ fixed to be 1 and makes O_2 compute exactly x_3. But this is impossible because x_3 is disconnected from O_2 by the two 1-paths. $\square$ $\square$

4.2 Multilective Upward Planar Circuits

In this section we consider the inversion complexity in multilective planar circuits. Since multilective upward planar circuits naturally generalize formulas, we have $I(f) \leq I_{M-UP}(f) \leq I_F(f) = d(f)$, where $I_{M-UP}(f)$ denote the minimum number of negations used for computing f by any multilective upward planar circuit. Next we show that multilective upward planar circuits are indeed more powerful than formulas; that is, relaxing the fan-out constraint will save some negation gates. By a Boolean function we will mean a single-output Boolean function.

Theorem 4.11 (Theorem 1.4 restated)**.** *For every Boolean function f, $I_{M-UP}(f) \leq \lceil \frac{d(f)+1}{2} \rceil$.*

Proof. The proof follows a similar line to that of Markov's. Recall that a connector of f_0 and f_1 is a function $\mu(y, y', x)$ satisfying that $\mu(i, 1-i, x) = f_i(x)$ for both $i = 0, 1$. Given a circuit C, we say a vertex (an input variable or a gate) in G is *out* if it lies in the outer-face of C. We say C is *out-negationed* if either C is monotone, or there is a negation gate in C which is out, and the sub-circuit below it is monotone (contains no other negation gates). We prove the following lemma about connectors.

Lemma 4.12. *For $i = 0, 1$, let $f_i(x)$ be a Boolean function which can be computed by a out-negationed multilective upward planar circuit containing t_i negation gates. In addition $t_1 = 1$. Then f_0 and f_1 have a connector $\mu(y, y', x)$ which can be computed by a multilective upward planar circuit containing at most* $\max\{t_0, t_1\}$ *negation gates and exactly one input node assigned with the variable y'. Moreover, this y'-node is out.*

of Lemma 4.12. For $i = 0, 1$, let C_i be the out-negationed multilective upward planar circuit computing f_i which contains t_i negation gates. If $t_0 = 0$, we can simply use $\mu = (y' \wedge f_0) \vee (y \wedge f_1)$, which satisfies all constraints. In the following we assume $t_0 \geq 1$.

Let $C = (y' \wedge C_0) \vee (y \wedge C_1)$. Let N_i be the out negation gate in C_i such that the sub-circuit below it, denoted by H_i, is monotone. We combine N_0 and N_1 into a new vertex N, and then identify it as the output gate of another circuit $(y' \wedge H_0) \vee (y \wedge H_1)$. Since N_i is out in C_i, the resulting circuit (still call it C) can be made multilective upward planar. Note that C contains two input vertices assigned y', both of which are out. Combining them together gives a multilective upward planar circuit containing $t_0 = \max\{t_0, t_1\}$ negation gates and exactly one copy of y', which is also out. It is easy to check that this circuit computes a connector of f_0 and f_1. $\square$ $\square$

Note that the proof no longer holds when $t_1 = 2$, since the circuit part $C_1 - H_1$ is not monotone in this case, and we need to use more negations.

We will prove a stronger result: For every Boolean function f, there is a out-negationed multilective upward planar circuit computing f which contains at most $\lceil \frac{d(f)+1}{2} \rceil$ negation gates. This is proved by induction on $d(f)$. When $d(f) \leq 1$, we simply use the formula construction [12], which uses $d(f) \leq \lceil\frac{d(f)+1}{2}\rceil$ negation gates. Next we assume that $d(f) \geq 2$, and the statement holds for all functions f' such that $d(f') < d(f)$.

Let $S \subseteq \{0,1\}^n$ be the collection of all inputs x such that every chain Y starting with x satisfies that $d_Y(f) \leq 1$. Then, for every chain Y ending at a vector $x \notin S$, it holds that $d_Y(f) \leq d(f) - 2$, since otherwise we can find a chain with decrease $\geq d(f) + 1$ by concatenating Y and the chain witnessing $x \notin S$. It is easy to see that ($x \in S$ and $x \leq y$) $\Rightarrow$ $y \in S$.

Define two functions $f_0(x)$ and $f_1(x)$ as follows:

$$f_0(x) = \begin{cases} 1 & \text{if } x \in S, \\ f(x) & \text{if } x \notin S, \end{cases}$$

and

$$f_1(x) = \begin{cases} f(x) & \text{if } x \in S, \\ 0 & \text{if } x \notin S. \end{cases}$$

Then we have $d(f_0) \leq d(f) - 2$ and $d(f_1) \leq 1$, and

$$f(x) = \begin{cases} f_1(x) & \text{if } x \in S, \\ f_0(x) & \text{if } x \notin S. \end{cases}$$

It is obvious that $f(x) = \mu(h_S(x), \overline{h_S(x)}, x)$, where $\mu(y, y', x)$ is any connector of f_0 and f_1. By the induction hypothesis and by Lemma 4.12, there exists a multilective upward planar circuit computing some connector μ which contains exactly one copy of y' and at most $\max\{\lceil(\frac{d(f)-1}{2}\rceil, 1\} = \lceil\frac{d(f)-1}{2}\rceil$ negation gates. If we replace every occurrence of y with a monotone formula computing $h_S(x)$, and replace the only copy of y' with a formula containing 1 negation gate that computes $\overline{h_S(x)}$, we will obtain a multilective upward planar circuit computing f which contains at most $\lceil\frac{d(f)+1}{2}\rceil$ negation gates. Note that by Lemma 4.12, the vertex assigned with y' is out. Therefore, the resulting circuit computing f is an out-negationed multilective upward planar circuit. This completes the proof of Theorem 4.11. □ □

References

[1] K. Amano and A. Maruoka. A superpolynomial lower bound for a circuit computing the clique function with at most (1/6)log log negation gates. *SIAM Journal of Computing*, 35(1):201–216, 2005.

[2] D. A. M. Barrington, C.-J. Lu, P. B. Miltersen, and S. Skyum. On Monotone Planar Circuits. In *Proceedings of the 14th Annual IEEE Conference on Computational Complexity (CCC)*, pages 24–31, 1999.

[3] M. Beynon and J. Buckle. On the planar monotone computation of boolean functions. *Theor. Comput. Sci.*, 53(2-3):267–279, 1987.

[4] T. Chakraborty and S. Datta. One-input-face MPCVP is hard for L, but in LogDCFL. In *Proc. of 26th FST TCS Conference*, 2006.

[5] M. Fischer. The complexity of negation-limited networks (a brief survey). *Lecture Notes in Computer Science*, 33:71–82, 1974.

[6] M. Jansen and J. Sarma M.N. Balancing Bounded Treewidth Circuits. In *Proeedings of CSR 2010 (To Appear)*, 2010.

[7] M. Karchmer and A. Wigderson. Monotone circuits for connectivity require super-logarithmic depth. In *STOC '88: Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 539–550, 1988.

[8] N. Limaye, M. Mahajan, and J. Sarma M.N. Upper bounds for monotone planar circuit value and variants. *Computational Complexity*, 18(3):377–412, 2009.

[9] A. A. Markov. On the inversion complexity of a system of functions. *J. ACM*, 5(4):331–334, 1958.

[10] I. L. Markov and Y. Shi. Simulating quantum computation by contracting tensor networks. *SIAM J. Comput.*, 38(3):963–981, 2008.

[11] W. F. McColl. On the planar monotone computation of threshold functions. In *Proceedings of 2nd annual symposium on theoretical aspects of computer science (STACS 85)*, pages 219–230, 1985.

[12] H. Morizumi. Limiting negations in formulas. In *Proceedings of 36th International Colloquium on Automata, Languages and Programming*, volume 5555 of *Lecture Notes in Computer Science*, pages 701–712. Springer, 2009.

[13] R. Raz and A. Wigderson. Monotone circuits for matching require linear depth. In *STOC '90: Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 287–292, 1990.

[14] A. A. Razborov. Lower bounds on the monotone complexity of some boolean functions. *Soviet Math. Dokl.*, 281:798–801, 1985.

[15] M. Santha and C. Wilson. Limiting negations in constant depth circuits. *SIAM J. Comput.*, 22(2):294–302, 1993.

[16] J. E. Savage. The performance of multilective vlsi algorithms. *Journal of Computer and System Science*, 29(2):243–273, 1984.

[17] S. C. Sung and K. Tanaka. Limiting negations in bounded-depth circuits: An extension of markovs theorem. In *Proceedings of International Symposium on Algorithms and Computation (ISAAC)*, volume 2906 of *Lecture Notes in Computer Science*, pages 108–116. Springer, 2003.

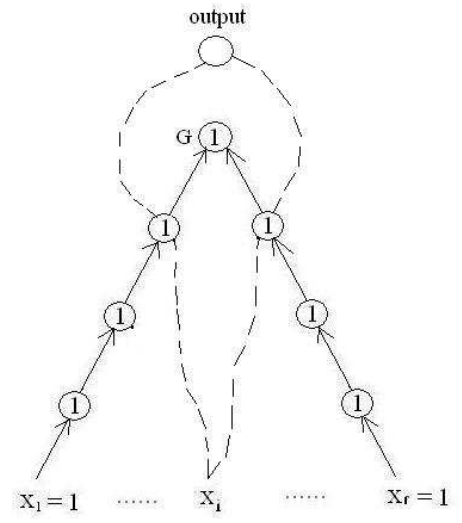

Figure 1: The variable x_i is insulated by the two 1-paths.

[18] E. Tardos. The Gap Between Monotone and Non-monotone Circuit Complexity is Exponential. *Combinatorica*, 7:393–394, 1987.

[19] L. G. Valiant. Short monotone formulae for the majority function. *J. Algorithms*, 5(3):363–366, 1984.

[20] H. Vollmer. *Introduction to Circuit Complexity: A Uniform Approach.* Springer New York Inc., 1999.

[21] H. Yang. An NC algorithm for the general planar monotone circuit value problem. In *Proc. 3rd IEEE Symp. on Parallel and Distributed Processing*, pages 196–203, 1991.

A Illustrative Figures

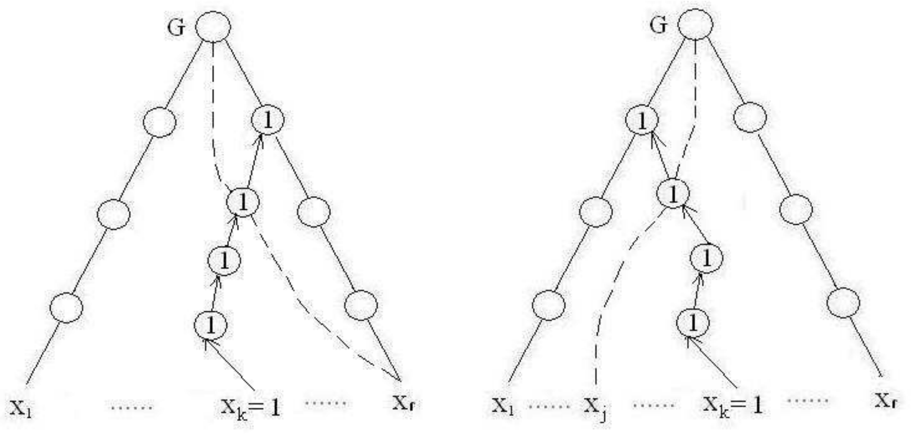

Figure 2: x_r and x_j are disconnected from G in respective case.

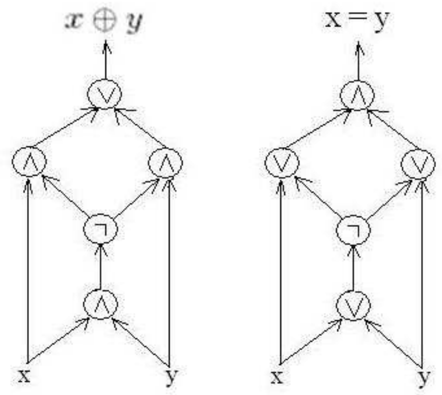

Figure 3: Planar circuits computing XOR and EQUIV

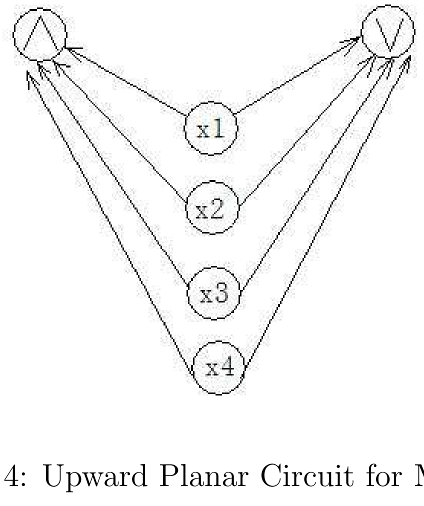

Figure 4: Upward Planar Circuit for MAXMIN

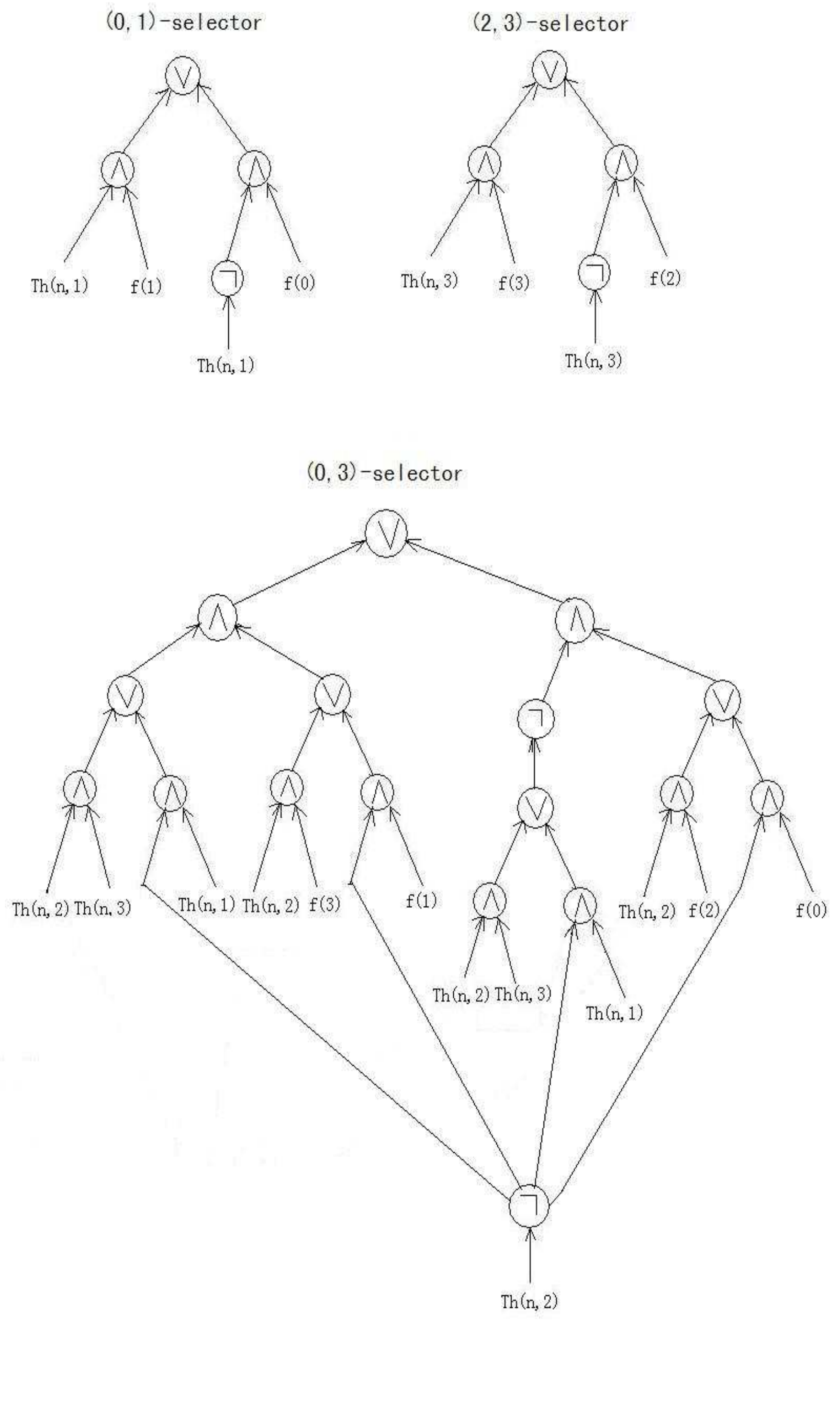

Figure 5: Replacement