$f_\alpha(x_1, x_2, \ldots, x_n) = f(y_1, y_2, \ldots, y_n)$ with

$$y_i := \begin{cases} \alpha(x_i) & \text{if } x_i \in U \\ x_i & \text{if } x_i \notin U. \end{cases}$$

In a natural way, an assignment α associates with a network β a subnetwork β_α, which is derived by fixing input variables according to α and eliminating the unnecessary gates.

In the sequel, we write $res(v)$ for $res_\beta(v)$ if β is kept fixed. $\# S$ denotes the cardinality of set S.

For proving the lower bound, we shall consider paths in a network. $(v \Rightarrow u)$ denotes a path from node v to node u. $(v \Rightarrow uE$ denotes a path from node v to a node in $pred(u)$.

Structure of the proof:

Step 1:

Make f independent of input variables which allow the elimination of three gates.

Step 2:

Prove for the remaining network without an inductive argument the existence of enough gates.

First, we shall describe Step 1:

Define for $1 \leq s \leq n$ the following statement E_s:

E_s: $C_{B_2}(f) \geq 3s-3$ for all functions

$$f: \{0,1\}^{n+3\log n+1} \longrightarrow \{0,1\}$$

with the property:

$[\exists S \subseteq \{1,2,\dots,n\}, \#S = s$ such that for $\sigma_1, \sigma_2, \sigma_3$ with $(\sigma_1), (\sigma_2), (\sigma_3) \in S$:

$$f(\sigma_1, \sigma_2, \sigma_3, \Gamma, x_1, \dots, x_n) = x^{\Gamma}_{(\sigma_1)} \wedge (x_{(\sigma_2)} \oplus x_{(\sigma_3)})].$$

E_1 is trivially true. (Note that $3 \cdot 1 - 3 = 0$)

Let $s \geq 2$ and assume that E_{s-1} is true.
Now we prove $E_{s-1} \Rightarrow E_s$.

Let β be any optimal B_2-network for f.
W.l.o.g., we can assume that for each $i \in S$ there is an unique node u in β with $op(u) = x_i$.

Case 1: $\exists i \in S$ such that $\#suc(x_i) \geq 3$.

By fixing x_i at 0 we can eliminate at least three gates of β. The reduced network computes the restriction $f_{x_i:=0}$ of f.

Inductive assumption $\Rightarrow$ $C_{B_2}(f_{x_i:=0}) \geq 3(s-1)-3$

$\Rightarrow$

$$C_{B_2}(f) \geq 3s-3.$$

Case 2: $\exists\, i \in S$ such that $\#\,suc(x_i) = 2$ and $\exists\, v \in suc(x_i)$ such that v is an $\wedge$-type gate.

Choose $c \in \{0,1\}$ such that $res(v)_{x_i := c}$ is constant. Then, by fixing x_i at c, we can eliminate all nodes in $suc(x_i)$ and all nodes in $suc(v)$.

Optimality of β $\Rightarrow$ These are at least three gates.

The reduced network computes the restriction $f_{x_i := c}$ of f.

Induction hypothesis $\Rightarrow$

$$C_{B_2}(f) \geq 3s - 3.$$

Case 3: $\neg$ (Case 1 or Case 2) and $\exists\, i \in S$ such that $\forall\, v \in suc(x_i)$, v is a $\oplus$-type gate.

We distinguish two subcases.

3.1 $\#\,suc(x_i) = 1$.

Then there exist nodes $u_1, u_2, \ldots, u_r$ in β with

1) $u_1 \in suc(x_i)$,
2) u_j is a $\oplus$-type gate for $1 \leq j \leq r$,
3) $u_{j+1} \in suc(u_j)$ and $\#\,suc(u_j) = 1$ for $1 \leq j < r$
4) $\#\,suc(u_r) > 1$ or for $w \in suc(u_r)$ there holds: w is an $\wedge$-type gate.

Let

x_i, g_1 input functions of gate u_1

$res(u_j), g_{j+1}$ input functions of gate u_{j+1}, $1 \leq j < r$.

Then there holds

$res(u_r) = x_i \oplus g$ where

$g = g_1 \oplus g_2 \oplus \dots \oplus g_r$ does not depend on x_i.

Note that $\mathcal{B}$ is acyclic.

$\Rightarrow$

$res(u_r)_{x_i := g}$ and $res(u_r)_{x_i := \neg g}$

are constant.

$\Rightarrow$

After the substitution of x_i by g or $\neg g$, we can eliminate the following gates:

- $u_1, u_2, \dots, u_r$ and all gates in $suc(u_r)$

g and $\neg g$, respectively can be computed using $r-1$ additional gates.

Two subcases are possible.

3.1.1 $\# suc(u_r) \geq 2$

Then we eliminate at least $r+2$ gates by fixing x_i at g or at $\neg g$.

3.1.2 $\#\,\mathrm{suc}(u_r) = 1$.

Then the unique $w \in \mathrm{suc}(u_r)$ is an $\wedge$-type gate. Choose $\tilde{g} \in \{g, \neg g\}$ such that

$$\mathrm{res}(w)_{x_i := \tilde{g}} \text{ is constant.}$$

$\Rightarrow$

After fixing x_i at $\tilde{g}$, we can eliminate at least $r+2$ gates, namely $u_1, u_2, \ldots, u_r$, w and all nodes in $\mathrm{suc}(w)$.

In both subcases, an application of the induction hypothesis gets

$$C_{B_2}(f) \geq 3s - 3.$$

3.2 $\#\,\mathrm{suc}(x_i) = 2$.

Then both successors u_1 and v_1 of x_i are $\oplus$-type gates.

Then there exists nodes $u_1, u_2, \ldots, u_r, v_1, v_2, \ldots, v_e \in B$ with

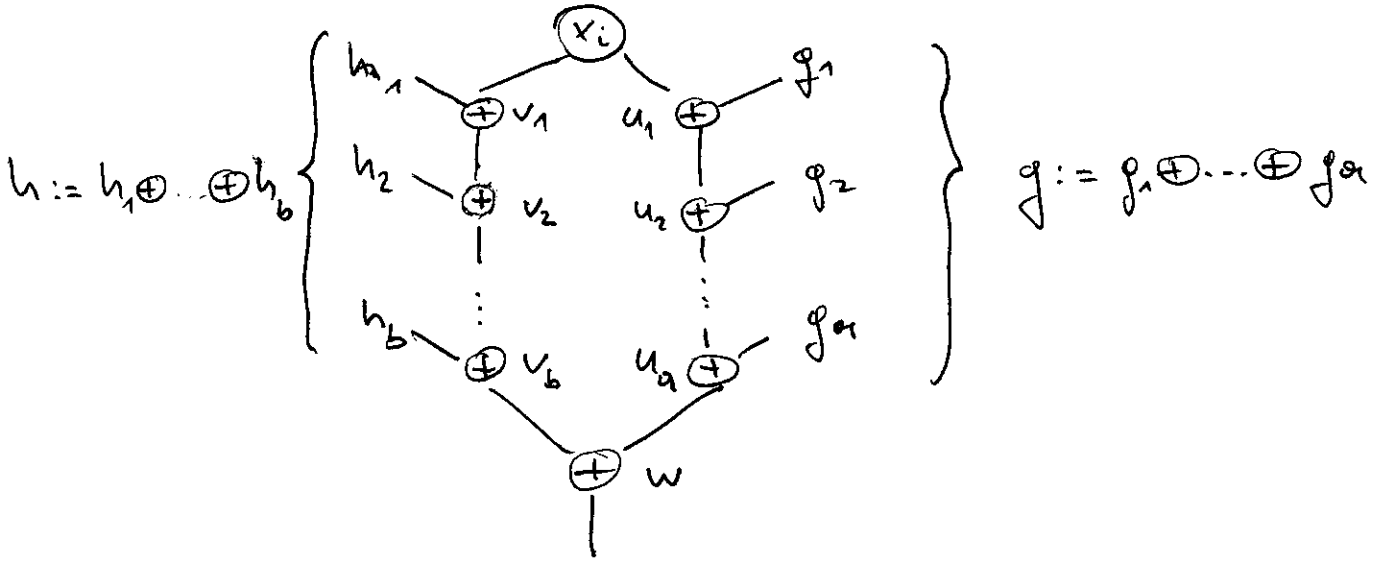
1) $u_1 \in \mathrm{suc}(x_i)$,
2) u_j is a $\oplus$-type gate for $1 \leq j \leq r$
3) $u_{j+1} \in \mathrm{suc}(u_j)$ and $\#\,\mathrm{suc}(u_j) = 1$ for $1 \leq j < r$
4) $\#\,\mathrm{suc}(u_r) > 1$ or $w \in \mathrm{suc}(u_r)$ is an $\wedge$-type gate

and

1) $v_1 \in Suc(x_i) \setminus \{u_1\}$
2) v_j is a $\oplus$-type gate for $1 \le j \le \ell$
3) $v_{j+1} \in Suc(v_j)$ and $\#Suc(v_j) = 1$ for $1 \le j < \ell$
4) $\#Suc(v_\ell) > 1$ or $w \in Suc(v_\ell)$ is an $\wedge$-type gate

Claim: $\{u_1, u_2, \dots, u_r\} \cap \{v_1, v_2, \dots, v_\ell\} = \emptyset$

Proof:
Assume that $\{u_1, u_2, \dots, u_r\} \cap \{v_1, v_2, \dots, v_\ell\} \neq \emptyset$.
Then we have the following situation:



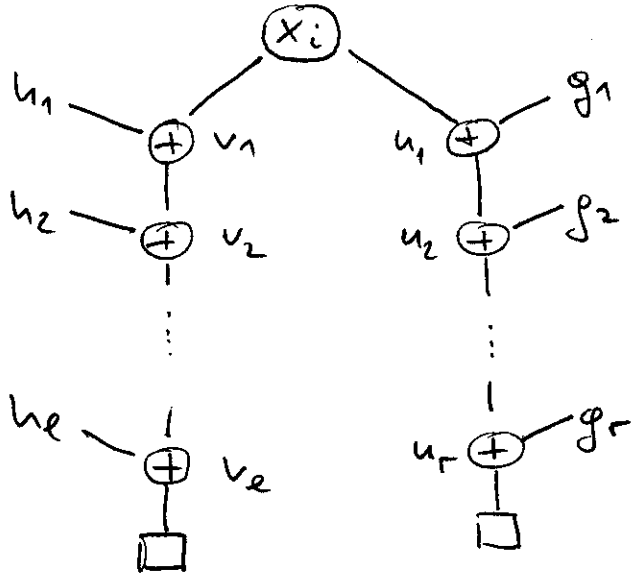
Then

$$res(w) = h \oplus g \oplus x_i \oplus x_i$$
$$= h \oplus g \oplus 0$$

$\Rightarrow$ f does not depend on x_i, a contradiction.

□

Therefore, we have the following situation.



β acyclic $\Rightarrow$

$g_1, g_2, \ldots, g_r$ do not depend on x_i

or

$h_1, h_2, \ldots, h_\ell$ do not depend on x_i

Exercise:

$\exists\, p, q$ such that h_p and also g_q depend on x_i $\Rightarrow$ β contains a cycle.

W.l.o.g., we can assume that $g_1, g_2, \ldots, g_r$ do not depend on x_i

Now we can prove $C_{B_2}(f) \geq 3s-3$ as in case 3.1.

Exercise:
Finish the proof of subcase 3.2.

Assume that none of the cases 1-3 arises. Then for all $i \in S$ the following holds

a) $\#\, Suc(x_i) = 1$. We denote the unique node in $Suc(x_i)$ by G_i.

b) G_i is an $\wedge$-type gate.

Let $G := \{ G_i \mid i \in S \}$.

Lemma 2.2

$\forall\, i, j \in S$ with $i \neq j$ there hold $G_i \neq G_j$.

Proof:

Suppose $\exists\, i, j \in S$, $i \neq j$, with $G_i = G_j$. Then there exists $c \in \{0,1\}$ such that

$$res(G_j)_{x_i := c} \text{ is constant.}$$

$\Rightarrow$

$$f_{x_i := c} \text{ does not depend on } x_j$$

But $f(\sigma_1, \sigma_2, \sigma_3, \tau, x_1, x_2, \ldots, x_n) = c \oplus x_j$ for

$(\sigma_1) \neq i, j$, $\tau = 1$, $x_{(\sigma_1)} = 1$,
$(\sigma_2) = i$, $x_i = c$ and $(\sigma_3) = j$

depends on x_j, a contradiction.

□

Case 4: $\exists\, i \in S$ such that $\#\, Suc(G_i) \geq 2$

Consider $c \in \{0,1\}$ with $res(G_i)_{x_i := c}$ is constant.

Then fixing x_i at c eliminates G_i and all gates in $\mathrm{suc}(G_i)$. There are at least three different such gates.

Hence, from the induction hypothesis it follows

$$C_{B_2}(f) \geq 3s - 3.$$

It remains to consider the following case.

Case 5: $\forall i \in S$ there holds $\#\mathrm{suc}(G_i) = 1$.

We denote the unique direct successor of G_i by Q_i.
Let

$$Q := \{ Q_i \mid i \in S \}.$$

Notations:

A path $(v \Rightarrow w)$ in β is called free, if no node $\neq v, w$ is in G.
A node w in β is called a split if the outdegree of w is at least two.
Let t be the node in β with $\mathrm{res}(t) = f$. A split w in β is called a free split if $\exists\, u_1, u_2 \in \beta$, $u_1 \neq u_2$ such that

a) $u_1, u_2 \in \mathrm{suc}(w)$,
b) $\exists$ free paths $(u_1 \Rightarrow t)$ and $(u_2 \Rightarrow t)$ in β.

A node w is called the collector of the free paths $(G_i \Rightarrow t)$ and $(G_j \Rightarrow t)$, $i \neq j$ if w is the first node which lies on both paths.

Lemma 2.3

$\forall i \in S$ there exists a free path $(G_i \Rightarrow t)$.

Proof:

Suppose that no free path $(G_i \Rightarrow t)$ exists.

$\Rightarrow$

Each path $(G_i \Rightarrow t)$ passes some G_j, $j \neq i$.

Construct the assignment α by fixing all variables except x_i such that

i) $res(G_v)_\alpha$ is constant $\forall v \in S \setminus \{i\}$,
ii) $f_\alpha = x_i^a$, $a \in \{0,1\}$.

Since each path $(G_i \Rightarrow t)$ goes through some G_v with $v \neq i$, $res(t)$ does not depend on x_i.

But this is a contradiction to

$$f_\alpha = x_i^a \text{ and } res(t)_\alpha = f_\alpha.$$

□

Note that Lemma 2.3 implies $G \cap Q = \emptyset$.

Lemma 2.4

Let $i, j \in S$, $i \neq j$. Let C be the collector of a free path $(G_i \Rightarrow t)$ and a free path $(G_j \Rightarrow t)$.

If $\nexists$ free split $\neq C$ on the path $(G_i \Rightarrow C)$ and $\nexists$ free split $\neq C$ on the path $(G_j \Rightarrow C)$ then the following holds:

i) C is a $\oplus$-type gate, and
ii) $\exists$ free path $(G_i \Rightarrow G_j)$ or $\exists$ free path $(G_j \Rightarrow G_i)$.

Proof:

Suppose that the assertion does not hold. We distinguish two cases.

Case 1: C is a $\oplus$-type gate.

Then by assumption, all paths $(G_i \Rightarrow G_j)$ and $(G_j \Rightarrow G_i)$ are not free.

Construct an assignment α by fixing all variables except x_i, x_j such that

a) $\text{res}(G_v)_\alpha$ is constant $\forall\, v \in \Sigma \setminus \{i,j\}$.
b) $f_\alpha = x_i \wedge x_j^a$, $a \in \{0,1\}$.

By assumption, all paths $(G_i \Rightarrow t)$ and $(G_j \Rightarrow t)$ go through C or a G_v, $v \in \Sigma \setminus \{i,j\}$.

Since there is no free path $(G_i \Rightarrow G_j)$ and no free path $(G_j \Rightarrow G_i)$, $\text{res}(G_i)_\alpha$ does not depend on x_j and $\text{res}(G_j)_\alpha$ does not depend on x_i. Hence,

$res(C)_\alpha = (x_i \oplus x_j)^b$, $b \in \{0,1\}$ or $res(C)_\alpha$ depends on at most one variable, and so the same holds for $res(t)_\alpha$.

$\Rightarrow$

$f_\alpha \neq res(t)_\alpha$, a contradiction.

Case 2: C is an $\wedge$-type gate.

Construct an assignment α by fixing all variables except x_i, x_j such that

a) $res(G_v)_\alpha$ is constant $\forall v \in S \setminus \{i,j\}$.
b) $f_\alpha = x_i \oplus x_j$.

(Here, we need the control variable r for forcing that such an assignment α exists.)

If $res(G_j)_\alpha$ depends on x_i, choose $c \in \{0,1\}$ such that $res(G_j)_{\alpha, x_i := c}$ is constant. Hence, $res(t)_{\alpha, x_i := c}$ does not depend on x_j, but $f_{\alpha, x_i := c} = c \oplus x_j$ depends on x_j, a contradiction.

The case $res(G_i)_\alpha$ depends on x_j is symmetric.

$\Rightarrow res(C)_\alpha = (x_i^a \wedge x_j^b)^c$, $a,b,c \in \{0,1\}$ or $res(C)_\alpha$ depends on at most one variable.

By assumption, all paths $(G_i \Rightarrow t)$ and $(G_j \Rightarrow t)$ go through C or a G_v, $v \in S \setminus \{i,j\}$.

Hence, for $res(t)_\alpha$ the same holds as for $res(C)_\alpha$.

$\Rightarrow$ $f_\alpha \neq res(t)_\alpha$, a contradiction. $\square$

Lemma 2.5

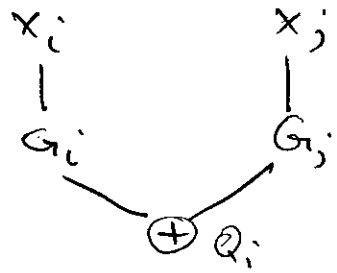
$\forall i,j \in S$ holds: $i \neq j \Rightarrow Q_i \neq Q_j$.

Proof:

Suppose that $\exists i,j$, $i \neq j$ but $Q_i = Q_j$.
Then Q_i is the collector of the free paths $(G_i \Rightarrow t)$ and $(G_j \Rightarrow t)$.

Lemma 2.4 $\Rightarrow$ Q_i is a $\oplus$-type gate.

Hence, we have the following situation.



Lemma 2.4 $\Rightarrow$

$\exists$ free path $(G_i \Rightarrow G_j)$ or $\exists$ free path $(G_j \Rightarrow G_i)$.

This is unpossible since both nodes G_i and G_j have outdegree one.

$\square$

Now, we have isolated 2s gates, namely the gates G_i, Q_i for $i \in S$.

Our goal is now to isolate $s-3$ further gates.

Lemma 2.6

There are at least $s-1$ mutually distinct splits in β.

Proof:

Let $S' = S$. Choose $i, j \in S'$ with free paths $(G_i \Rightarrow t)$ and $(G_j \Rightarrow t)$ such that

- $\forall\, \ell \in S' \setminus \{i,j\}$ no free path $(G_\ell \Rightarrow t)$ goes through the collector C of the free paths $(G_i \Rightarrow t)$ and $(G_j \Rightarrow t)$.

Exercise:
Show that such a choice is possible.

Lemma 2.4 $\Rightarrow$

One of the paths $(G_i \Rightarrow C[$ and $(G_j \Rightarrow C[$, respectively splits into a free path to the output node t or splits into a free path to the node G_j and G_i, respectively.

W.l.o.g., let the path $(G_i \Rightarrow C[$ split.

Set $$S' := S' \setminus \{i\}.$$

Construction $\Rightarrow$

No free path $(G_\ell \Rightarrow t)$, $\ell \in S'$ has a common

node with the path $(G_i \Rightarrow C[$.

Repeating this argument $s-1$ times proves the lemma.

□

Since we have at least $s-1$ splits, we have to connect at least $2(s-1)$ edges with the output node t.

For edges on free paths, no node in G can be used to connect these with t by the definition of a free path.

Next, we shall prove that all but one of the $s-1$ splits from Lemma 2.6 have to be free.

Assume that less than $s-1$ splits isolated in the proof of Lemma 2.6 are free.

Lemma 2.4 $\Rightarrow$

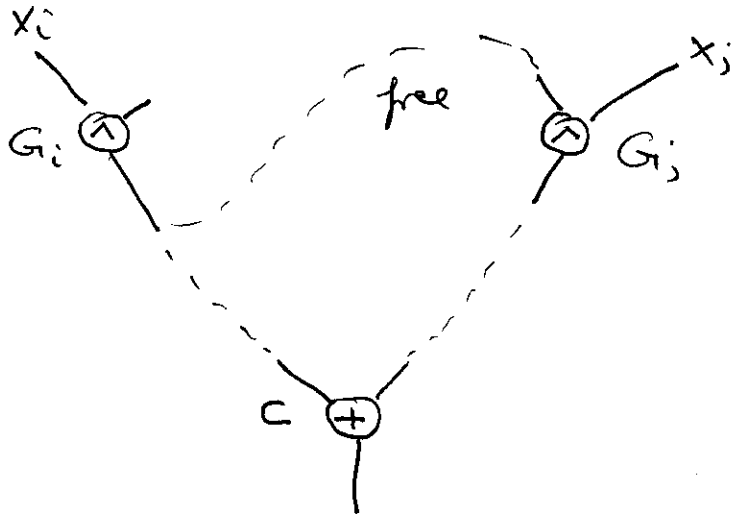
$\exists\, i, j \in S$, $i \neq j$ with the following property:

- Let C be the collector of the free paths $(G_i \Rightarrow t)$ and $(G_j \Rightarrow t)$.

Then there is no free split on the path $(G_i \Rightarrow C[$ and no free split on the path $(G_j \Rightarrow C[$.

There is a free path $(G_i \Rightarrow G_j)$ and C is a $\oplus$-type gate.

Then we have the following situation:



Lemma 2.7

$\forall\, v \in S \setminus \{j\}$, $\exists$ free path $(G_v \Rightarrow G_i)$ or $\exists$ free path $(G_v \Rightarrow G_j)$.

Proof:

For i we know by assumption that $\exists$ free path $(G_i \Rightarrow G_j)$.

Assume that $\exists\, \ell \in S \setminus \{i,j\}$ with

$\nexists$ free path $(G_\ell \Rightarrow G_i)$ and $\nexists$ free path $(G_\ell \Rightarrow G_j)$.

Now we construct an assignment α by fixing all variables except x_i, x_j, x_ℓ such that

a) $\operatorname{res}(G_v)_\alpha$ is constant $\forall\, v \in S \setminus \{i,j,\ell\}$

b) $f_\alpha = x_j \wedge (x_i \oplus x_\ell)$.

We distinguish two cases.

Case 1: $\mathrm{res}(G_j)_\alpha$ does not depend on x_i.

Now we fix x_ℓ at $a \in \{0,1\}$ such that

$\mathrm{res}(G_\ell)_{\alpha, x_\ell := a}$ is constant.

$\Rightarrow$

$$f_{\alpha, x_\ell := a} = x_j \wedge x_i^b \quad , b \in \{0,1\}.$$

Now, as in the proof of Lemma 2.4, we show that Case 1 cannot happen.

Case 2: $\mathrm{res}(G_j)_\alpha$ depends on x_i.

Then, $\mathrm{res}(G_j)_\alpha = (x_i^c \wedge x_j^d)^e$, $c,d,e \in \{0,1\}$.

Fix x_i at $\neg c$.

Then $\mathrm{res}(G_j)_{\alpha, x_i := \neg c}$ is constant.

$\Rightarrow$

$\mathrm{res}(\epsilon)_{\alpha, x_i := \neg c}$ does not depend on x_j.

But

$$f_{\alpha, x_i := \neg c} = x_j \wedge (\neg c \oplus x_\ell) = x_j \wedge x_\ell^c$$

depends on x_j, a contradiction.

□

Now we shall prove that all other splits isolated in the proof of Lemma 2.6 have to be free.

Lemma 2.8

For all $\ell, v \in S \setminus \{t\}$, $v \neq \ell$ the following holds.
If D is the collector of a free path $(G_\ell \Rightarrow t)$ and a free path $(G_v \Rightarrow t)$ then
$\exists$ free split $\neq D$ on path $(G_\ell \Rightarrow D)$ or
$\exists$ free split $\neq D$ on path $(G_v \Rightarrow D)$.

Proof.

Suppose that
$\nexists$ free split $\neq D$ on path $(G_\ell \Rightarrow D)$ and
$\nexists$ free split $\neq D$ on path $(G_v \Rightarrow D)$.

Lemma 2.4 $\Rightarrow$
There is a free path $(G_\ell \Rightarrow G_v)$ or there is a free path $(G_v \Rightarrow G_\ell)$. and D is a $\oplus$-type gate.

Hence, we can apply Lemma 2.7 to ℓ and v.

$\Rightarrow$
$\exists$ a path $(G_j \Rightarrow G_\ell)$ or $\exists$ a path $(G_j \Rightarrow G_v)$.

Note that by construction $\exists$ path $(G_i \Rightarrow G_j)$.
Furthermore, by construction, there exist paths $(G_\ell \Rightarrow G_j)$ and $(G_v \Rightarrow G_j)$.

Hence, we have a cycle in the network. But this cannot happen by the definition of a network.

□

From Lemma 2.8 we can directly derive the following lemma.

Lemma 2.9

There are at least $s-2$ mutually distinct free splits in β.

Exercise

Prove Lemma 2.9.

Now we can count the gates in β.

Lemma 2.9 and Lemma 2.3 $\Rightarrow$

We have to connect at least $2(s-2)+2$ edges on free paths to the output node t.

Since the paths are free, the nodes in G cannot help to connect these edges to the output node.

For the nodes in Q only one input wire is free for connecting these edges. There are s nodes in Q. Hence, at least $s-2$ edges have to connect to the output node using new nodes.

One new node (except the output node) can decrease the number of edges only by one. The output node can decrease the number of edges only by two.

$\Rightarrow$

We need at least $s-3$ new gates.

Altogether, we have obtained

$$C_{B_2}(f) \geq \#G + \#Q + s - 3 = 3s - 3.$$

This concludes the proof of the following theorem

Theorem 2.4

Let $f: \{0,1\}^{n+3\log n+1} \to \{0,1\}$ be defined by

$$f(a_1, a_2, \ldots, a_{3\log n}, r, x_1, x_2, \ldots, x_n) = x^r_{(a_1)} \wedge (x_{(a_2)} \oplus x_{(a_3)}).$$

Then $C_{B_2}(f) \geq 3n - 3$.

Magnus Gausdal Find, Alexander Golovnev, Edward A. Hirsch, Alexander S. Kulikov

A better-than-3n lower bound for the circuit complexity of an explicit function, ECCC, Revision of Report No. 166 (2015), February 16, 2016 (37 pages)

have shown a $(3 + \frac{1}{86})n - o(n)$ lower bound for another function by a much more complicated proof.

Note that $x \oplus y = x \wedge \bar{y} \vee \bar{x} \wedge y$. Hence, each $\oplus$-type gate can be realized using three gates from $\{\wedge, \vee\}$ and some negations. Therefore, if we restrict us to the base Ω_0, we can realize a B_2-network by an Ω_0-network increasing

the size of the network at most by the constant factor three. Hence, for proving a non-linear lower bound for the network complexity of a Boolean function, it suffices to prove this with respect to the base Ω_0. Such a non-linear lower bound would imply a non-linear lower bound for the B_2-complexity of the same function as well.

But also with respect to the base Ω_0, only linear lower bounds with small constant factor are proved for explicit defined Boolean functions. The largest lower bound with complete proof is $4n$ by Uri Zwick. In

Kazuo Iwama, Oded Lachish, Hiroki Morizumi, Ran Raz, An explicit lower bound of $5n - o(n)$ for Boolean circuits, preprint 2005. (Can be obtained via the homepage of Ran Raz.),

a $5n$ lower bound is claimed. The proof ideas look fine. But too much is left for the reader such that the proof cannot be considered to be complete.

The inability to prove lower bounds for the Ω_0-complexity of explicit Boolean functions has led to the consideration of restricted models of Boolean networks like monotone or bounded-depth Boolean networks. For both

restricted models, exponential lower bounds for the complexity of explicit Boolean functions are known. The hope is that methods, developed for a restricted model, could be extended to get a proof of a super-linear lower bound for the Ω_0-complexity of an explicit Boolean function.

It is my opinion that methods developed with respect to bounded-depth networks do not help to get a solution of the general problem. In contrast to this, I believe that methods developed with respect to monotone networks could help. Hence, we shall consider monotone Boolean networks extensively.

3. Monotone Boolean networks

Let $\Omega_m := \{\wedge, \vee\}$ denote the monotone base. An Ω_m-network is also called monotone network. First, we shall mention some fundamental properties of monotone Boolean functions and monotone networks.

Lemma 3.1

Each prime implicant of a monotone Boolean function $f \in M_n$ contains only non-negated variables.

Proof:

exercise

■

Exercise

Show that both constant functions are monotone.

Theorem 3.1

The set of non-constant monotone functions and the set of functions which can be computed by a monotone network are equal.

Proof:

exercise

■

By an estimation of the number of n-ary monotone Boolean functions and an application of Shannon's counting argument, one can also prove that nearly all monotone Boolean functions have exponential (monotone) network complexity (see Wegener, pp. 98 - 105). But with respect to single output monotone Boolean functions, no technique for counting a super-linear number of gates has been developed before 1985. The best lower bound for the monotone network complexity of an explicit function in M_n before 1985 was of size $4n$.

Jürgen Tiekenheinrich, A 4n-lower bound on the monotone network complexity of a one-output Boolean function, IPL 18 (1984), 201-202.

For functions in $M_{n,m}$ where $m = \Theta(n)$, non-linear lower bounds have been proved since 1969. All these lower bound proofs use the following property of each monotone network β which computes a function $f = (f_1, f_2, \dots, f_m) \in M_{n,m}$.

For all node g in β, the function $res_\beta(g)$ can be written as a DNF-formula; i.e.,

$$res_\beta(g) = \bigvee_{j=1}^{t} m_j ,$$

where each m_j is a monomial. Starting at the input nodes, we can compute these DNF-formulas in the obvious way by applying the properties of the Boolean operations.

Let u_i, $1 \le i \le m$ be the output node of the network β which computes the function f_i. For $res_\beta(u_i) = \bigvee_{j=1}^{t_i} m_j$, the following hold:

1) For $1 \le j \le t_i$, the monomial m_j is an implicant of the function f_i.
2) For all prime implicants p of f_i there is a $j \in \{1, 2, \dots, t_i\}$ with $m_j = p$.

If the first property is not fulfilled then there

is an input $(a_1, a_2, \ldots, a_n) \in \{0,1\}^n$ such that

$$f_i(a_1, a_2, \ldots, a_n) = 0 \quad \text{but} \quad res_\beta(u_i)(a_1, a_2, \ldots, a_n) = 1.$$

If the second property is not fulfilled then there is an input $(a_1, a_2, \ldots, a_n) \in \{0,1\}^n$ such that

$$f_i(a_1, a_2, \ldots, a_n) = 1 \quad \text{but} \quad res_\beta(u_i)(a_1, a_2, \ldots, a_n) = 0.$$

Most functions considered for proving lower bounds are homogeneous. A Boolean function $f \in M_{n,m}$ is called <u>k- homogeneous</u> if all prime implicants of f are of length k.

Before we consider explicitly defined monotone functions in $M_{n,m}$, we shall develop some <u>replacement rules</u>. The idea is to replace a gate by another gate which computes a certain function without changing the function computed by the whole network.

<u>Reference</u>

- <u>Kurt Mehlhorn</u>, <u>Zvi Galil</u>, Monotone switching circuits and Boolean matrix product, Computing 16 (1976), 99 - 111.

For $f, g \in B_n$, we define

$$f \leq g :\Leftrightarrow f \wedge g = f.$$

Then f is called a <u>subfunction</u> of g.

Theorem 3.2

Let $f, g \in M_n$ and $t \in PIM(g)$ where $tt' \notin PIM(f)$ for all monomials t' (including the empty monomial). Let h be defined by $PIM(h) = PIM(g) \setminus \{t\}$. If there is a gate v computing the function g in a monotone network $\mathcal{B}$ which computes the function f and we replace in $\mathcal{B}$ the gate v by a gate v' which computes h then the resulting network $\mathcal{B}'$ still computes f.

Proof:

Let f' be the function computed by the network $\mathcal{B}'$. Since $h \leq g$ there holds $f' \leq f$.

Assume that $f' \neq f$.

Then there is $a \in \{0,1\}^n$ such that $f'(a) = 0$ but $f(a) = 1$.

Since we have obtained $\mathcal{B}'$ from $\mathcal{B}$ by the replacement of the gate v in $\mathcal{B}$ by the gate v', there holds

$$h(a) = 0 \quad \text{and} \quad g(a) = 1.$$

By construction, $g = h \vee t$. Hence, $t(a) = 1$.

Consider $t^* \in PIM(f)$ with $t^*(a) = 1$.
Since $f(a) = 1$, such a prime implicant of f exists.

Claim: $\exists$ monomial t' with $t \cdot t' = t^*$.

Proof of claim:

Consider any variable x_i in t.

$t(a) = 1 \Rightarrow a_i = 1$.

Consider the input $b \in \{0,1\}$, defined by

$$b_j := \begin{cases} a_j & \text{if } j \neq i \\ 0 & \text{if } j = i \end{cases}$$

Then by definition

$$b < a \text{ and } t(b) = 0.$$

$\Rightarrow$

$$f(b) = f'(b) = 0.$$

$\Rightarrow$

$$t^*(b) = 0$$

Since b and a differ only at position i, x_i is a variable in t^*.

Since x_i has been chosen to be any variable in t it follows that t is a submonomial of t^*.

$\Rightarrow$ $\exists\ t'$ with $tt' = t^*$

□

By the choice of t, t is not a submonomial of any prime implicant of f, a contradiction. ∎

Theorem 3.3

Let β be a monotone network computing f and let v be a gate in β with $res_\beta(v) = g$. Let t, t_1 and t_2 be monomials such that

i) $tt_1, tt_2 \in IM(g)$ and
ii) $\forall$ monomials t': $t'tt_1, t'tt_2 \in IM(f) \Rightarrow t't \in IM(f)$.

If we replace in β the gate v by a gate v' which computes $h = g \vee t$ then the resulting network β' still computes f.

Proof:

Let f' be the function computed by β'.

$g \leq h \Rightarrow f \leq f'$.

Suppose that $f' \neq f$. $\Rightarrow$

$\exists\, a \in \{0,1\}^4 : f'(a) = 1$ and $f(a) = 0$.

$\Rightarrow$

$h(a) = 1$, $g(a) = 0$ and $t(a) = 1$.

Consider $t^* \in PIM(f')$ with $t^*(a) = 1$.

$f(a) = 0 \Rightarrow t^*t \notin IM(f)$.

To get a contradiction, it suffices to prove

$t^*tt_1, t^*tt_2 \in IM(f)$.

Consider $b \in \{0,1\}^n$ with $t^*tt_j(b) = 1$ $j = 1,2$.

$t^* \in PIM(f') \Rightarrow f'(b) = 1$

$tt_j \in IM(g) \Rightarrow g(b) = 1$

Hence,

$g \leq h \Rightarrow h(b) = 1$.

$\Rightarrow$

Since at the only gate where β and β' differ, the same value is computed with input b, both networks β and β' compute the same value with respect to the input b.

$\Rightarrow$ $f(b) = f'(b) = 1 \Rightarrow t^*tt_j \in IM(f)$, $j = 1,2$.

■

Now, we shall consides explicit functions in $M_{n,m}$.