The complexity of Boolean functions

References

[1] Ingo Wegener, The Complexity of Boolean Functions, Teubner - Wiley, 1987.

[2] Paul E. Dunne, The Complexity of Boolean Networks, Academic Press, 1988.

[3] Ravi B. Boppana, Michael Sipser, The Complexity of Finite Functions, in Handbook of Theoretical Computer Science, Vol. A, Algorithms and Complexity, MIT-Press and Elsevier, 759 - 804, 1990.

[4] John E. Savage, Models of Computation: Exploring the Power of Computing, Addison-Wesley, 1998.

[5] Peter Clote, Evangelos Kranakis, Boolean Functions and Computation Models, Springer, 2002.

[6] Steven Rudich, Avi Wigderson (eds.), Computational Complexity Theory, IAS / Park City Mathematical Series, Vol. 10, AMS, 2004.

[7] Sanjeev Arora, Boaz Barak, Computational Complexity: A Modern Approach, Cambridge University Press, 2009.

[8] Norbert Blum, On Negations in Boolean Networks, LNCS 5760, 18 - 29, 2009.

[9] Stasys Jukna, Boolean Function Complexity: Advances and Frontiers, Springer, 2012.

0. Motivation

Is P = NP ? This means, is deterministic polynomial time equal nondeterministic polynomial time ?

The P ? NP - question is the most famous open problem in computer science.

A popular approach to attack the P ? NP - problem is a consequence of the following theorem.

Theorem 0.1

If a language $L \subseteq \{0,1\}^*$ can be accepted by a one-tape deterministic Turing machine in time $T(n)$ then for each $n \in \mathbb{N}$ there exists a Boolean circuit of size $O(T^2(n))$ which computes the characteristic function of $L \cap \{0,1\}^n$.

Proof:

Not hard. See for example [4], pp. 124 - 127. ∎

Remark:

Using binary coding, each language $L \subseteq \Sigma^*$, Σ finite alphabet, can be identified with a language $L' \subseteq \{0,1\}^*$.

Theorem 0.1 $\Rightarrow$

The proof of a nonpolynomial lower bound for the circuit complexity of the characteristic function of a language $L \in NP$ implies $P \neq NP$

Main topic of the lecture

The development of methods for proving lower bounds for Boolean functions.

1. Preliminaries and asymptotic results

$B_{n,m} := \{ f \mid f: \{0,1\}^n \to \{0,1\}^m \}$ is the set of all n-ary Boolean functions with m outputs. Instead of $B_{n,1}$ we write B_n.

$x_i \in B_n$, $1 \leq i \leq n$ is the i-th variable.

Let

$V_n := \{x_i \mid 1 \leq i \leq n\}$ and $V_n' := V_n \cup \{\bar{x}_i \mid 1 \leq i \leq n\}$.

Variables and negated variables are called literals. A function $m: \{0,1\}^n \to \{0,1\}$ which is the product of some literals is called a monomial. The empty product is

the constant function 1.

For $f, g \in B_n$ we define

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

In that case f is a <u>subfunction</u> of g.

$$IM(f) := \{ t \mid t \text{ monomial}, t \leq f \}$$

is the set of <u>implicants</u> of the function f.

An implicant $t \in IM(f)$ is a <u>prime implicant</u> of f if $\forall\, t' \in IM(f): [t \leq t' \leq f \Rightarrow t = t']$.

$PIM(f) \subseteq IM(f)$ is the set of all prime implicants of f.

Let $a := (a_1, a_2, \ldots, a_n)$, $b := (b_1, b_2, \ldots, b_n)$.
We write $a \leq b$ iff $a_i \leq b_i$ for $1 \leq i \leq n$.

A function $f = (f_1, f_2, \ldots, f_m) \in B_{n,m}$ is <u>monotone</u> iff for all $a, b \in \{0,1\}^n$, $a \leq b$ implies $f_i(a) \leq f_i(b)$, $1 \leq i \leq m$.

$M_{n,m}$ denote the set of monotone functions in $B_{n,m}$.
We also write M_n for $M_{n,1}$.

B_2 is the set of <u>basic operations</u>. Let $\Omega \subseteq B_2$.

An <u>Ω-network</u> $\mathcal{B}$ is a directed, acyclic graph such that

a) each node has indegree ≤ 2.

b) The nodes u with indegree 0 are input nodes and are labelled with $op(u) \in V_n$.

c) Each non-input node u is labelled by an $op(u) \in \Omega$.

A node with outdegree 0 is an <u>output node</u>.

For a node u in β let

$$suc(u) := \{ v \mid u \to v \text{ is an edge in } \beta \}$$

and

$$pred(u) := \{ v \mid v \to u \text{ is an edge in } \beta \}.$$

be the sets of direct successors and direct predecessors.

With each node u, we associate a function $res_\beta(u) : \{0,1\}^n \to \{0,1\}$ (n is the number of input nodes of β where

$$res_\beta(u) := \begin{cases} op(u) & \text{if } u \text{ is an input node} \\ \neg\, res_\beta(v) & \text{if } op(u) = \neg, \text{ where } v \text{ is the direct predecessor of } u \\ res_\beta(v)\ op(u)\ res_\beta(w) & \text{otherwise, where } v, w \text{ are the direct predecessors of } u \end{cases}$$

For $u \in \mathcal{B}$, $res_{\mathcal{B}}(u)$ is the function computed at node u in $\mathcal{B}$. Let $G \subset \mathcal{B}_n$.

The minimal number of gates in an Ω-network which computes G is the $\underline{\Omega\text{-complexity}}$ $C_\Omega(G)$ of G. Usually, $\neg$-gates are not counted

For $f \in \mathcal{B}_n$ and $a \in \{0,1\}$ let

$$f^a := \begin{cases} f & \text{if } a = 1 \\ \neg f & \text{if } a = 0 \end{cases}$$

$f \in \mathcal{B}_2$ is $\underline{\wedge\text{-type}}$ if $\exists\, a,b,c \in \{0,1\}: f(x,y) = (x^a \wedge y^b)^c$

$f \in \mathcal{B}_2$ is $\underline{\oplus\text{-type}}$ if $\exists\, a \in \{0,1\}: f(x,y) = (x \oplus y)^a$

A node $u \in \mathcal{B}$ with $op(u)$ is $\wedge$-type ($\oplus$-type) is called $\underline{\wedge\text{-type-gate}}$ ($\underline{\oplus\text{-type-gate}}$).

Lemma 1.1

The functions $f \in \mathcal{B}_2$ can be classified in the following way. There are

i) two constant functions,
ii) four functions depending on one variable,
iii) ten functions depending on two variables, eight of these functions are $\wedge$-type and two are $\oplus$-type.

Proof:

exercise

■

Exercise

Show that $|B_n| = 2^{2^n}$ and $|B_{n,m}| = 2^m 2^n$

Each Ω-network β can also be viewed as a straight line program $g_1, g_2, \dots, g_t$ of $t \geq n$ Boolean functions such that the first n functions are input variables $g_1 = x_1, g_2 = x_2, \dots, g_n = x_n$, and each subsequent function g_i is $g_i = \neg g_j$ with $j < i$ or $g_i = g_{i_1}\, op(g_i)\, g_{i_2}$ where $i_1, i_2 < i$ and $op(g_i) \in \Omega \setminus \{\neg\}$.

We also use synonymous circuit for network.

Goal:
The development of lower and upper bounds for not explicid defined functions in B_n.

Let

$$\phi(n,t) := \{ f \in B_n \mid C_{\Omega_0}(f) \leq t \}$$

First we shall eshimate an upper bound for $\phi(n,t)$.

Lemma 1.2

$$\phi(n,t) \leq t^t \cdot g^t \cdot e^{2n}$$

Proof:

Obviously, $\phi(n,t)$ is upper bounded by

the number of distinct Ω_0-networks with $\leq t$ gates and n input nodes.

Next we wish to derive an upper bound for the number of distinct circuits with exactly $t' \leq t$ gates and n input nodes.

For each gate there are

- $|\Omega_0| = 3$ possibilities to choose its type and
- $\leq t' - 1 + n$ possibilities to choose each of its predecessors.

Furthermore, there are $t'!$ permutations of the indices of the gates leading to different descriptions of the same network. Hence, the number of distinct circuits with t' gates and n input nodes is upper bounded by

$$\frac{3^{t'} (t'+n)^{2t'}}{t'!}$$

$\Rightarrow$

$$\phi(t,n) \leq t \cdot \frac{3^t (t+n)^{2t}}{t!}$$

Note that

$$(t-1)! \geq \left(\frac{t}{3}\right)^t \quad \text{and} \quad 1+x \leq e^x.$$

Exercise:
Prove $(t-1)! \geq \left(\frac{t}{3}\right)^t$.

Hence,

$$\phi(t,n) \leq \frac{g^t (t+n)^{2t}}{t^t}$$

$$= g^t \cdot t^t \left(1 + \frac{n}{t}\right)^{2t}$$

$$\leq t^t \cdot g^t \cdot e^{2n}$$

■

To compute all functions in B_n by an Σ_0-network with at most t gates, we need

$$\phi(t,n) \geq |B_n| = 2^{2^n}.$$

If t is chosen in such a way, Lemma 1.2 implies

$$t^t \cdot g^t \cdot e^{2n} \geq 2^{2^n}$$

$$\Rightarrow 2^{t \cdot \log t} \cdot 2^{4t} \cdot 2^{4n} \geq 2^{2^n}$$

$$\Leftrightarrow t \cdot \log t + 4t + 4n \geq 2^n$$

Assume $t \leq \frac{2^n}{n}$. Then we obtain

$$t \cdot \log t + 4t + 4n$$

$$\leq \frac{2^n}{n} \cdot (n - \log n + 4) + 4n$$

$$= 2^n - \frac{(\log n - 4)}{n} 2^n + 4n$$

But for $n > 32$, we obtain

$$4n - \frac{\log n - 4}{n} \cdot 2^n < 0.$$

Hence, t has to be larger than $\frac{2^n}{n}$ for $n > 32$.

This proves that there are Boolean functions in B_n which need more than $\frac{2^n}{n}$ gates. To prove that almost all functions in B_n need more than $\frac{2^n}{n}$ gates, let

$$a(n) := 2^n - 2^n \cdot n^{-1} \cdot \log\log n.$$

Let $B_n^* \subset B_n$ be the set of those $2^{a(n)}$ functions in B_n which use the smallest number of gates.

In the same way as above, we can prove that for n large enough, B_n^* contains also a function which need more than $\frac{2^n}{n}$ gates.

Exercise: (for n large enough)
Prove that B_n^* contains a function which need more than $\frac{2^n}{n}$ gates.

By the definition of B_n^*, all functions in $B_n \setminus B_n^*$ need more than $\frac{2^n}{n}$ gates. Altogether, we have proven the following theorem.

Theorem 1.1

For sufficiently large n at least

$$\left(1 - 2^{-2^n n^{-1} \log\log n}\right) \cdot |B_n|$$

of the $|B_n|$ functions in B_n need more than

$\frac{2^n}{n}$ gates.

Using the disjunctive normal form of a given Boolean function, it is easy to see that each function in B_n can we computed by an Ω_0-network which has at most $n \cdot 2^n + n$ gates.

We shall show that always $\frac{2^n}{n} + o(\frac{2^n}{n})$ gates suffices. First we shall present two less efficient networks.

a) The decoding network:

Consider any $f \in B_n$. Let $f_0 \in B_{n-1}$ and $f_1 \in B_{n-1}$ those subfunctions of f which we obtain if we fix $x_n := 0$ and $x_n := 1$, respectively. Then we can write

$$(*) \qquad f = (\overline{x_n} \wedge f_0) \vee (x_n \wedge f_1).$$

Note that for the construction, we have to negate the same variable at most once. For doing this, n negations suffice. These are added at the end of the construction.

Let $C_{\Omega_0}(B_n)$ denote the minimum number of gates without negations of input nodes which suffices to realize any function in B_n.

Then we obtain for $n > 2$

$$C_{\Omega_0}(B_n) \leq 2 \cdot C_{\Omega_0}(B_{n-1}) + 3.$$

Note that

$$C_{\Omega_0}(B_2) = 3.$$

$\Rightarrow$

$$C_{\Omega_0}(B_n) \leq 3 \cdot 2^{n-2} + 3 \cdot \sum_{i=0}^{n-3} 2^i$$

Note that $\sum_{i=0}^{n-3} 2^i = \frac{1-2^{n-2}}{1-2} = 2^{n-2} - 1.$

Hence, we obtain

$$C_{\Omega_0}(B_n) \leq 3 \cdot 2^{n-2} + 3 \cdot 2^{n-2} - 3$$

$$= 3 \cdot 2^{n-1} - 3$$

□

This improves the obvious upper bound by the factor n.

b) <u>The improvement of the decoding network:</u>

<u>Observation:</u>

The decoding network computes all used 2^{n-3} subfunctions in B_3 by disjoint subnetworks. It would be more efficient to compute all 2^8 functions in B_3 in advance and use to

use them if needed.

Let $C^*_{\Omega_0}(B_k)$ be the minimum number of of gates to compute all functions in B_k where the negations of input gates are not counted. Then we obtain because of (*)

$$C^*_{\Omega_0}(B_k) \leq C^*_{\Omega_0}(B_{k-1}) + 3 \cdot |B_k|.$$

Exercise:
Show that $C^*_{\Omega_0}(B_2) \leq 12$.

Since $C^*_{\Omega_0}(B_2) \leq 16$, we obtain

$$\begin{aligned} C^*_{\Omega_0}(B_k) &\leq 3 \cdot 2^{2^k} + 3 \cdot \left(\sum_{i=2}^{k-1} |B_i| \right) \\ &= 3 \cdot 2^{2^k} + 3 \cdot \sum_{i=2}^{k-1} 2^{2^i} \\ &\leq 3 \cdot 2^{2^k} + 6 \cdot 2^{2^{k-1}} \end{aligned}$$

The computation of the function f can be realized as follows.

(1) Choose the appropriate k.
(2) Compute all functions in B_k.
(3) Apply the decoding network to compute f.

Since all necessary subfunctions in B_k are already computed the construction of the decoding network for f can be terminated after $n-k$ steps.

The used number of gates in the decoding network is

$$\leq 3 \cdot 2^{n-k} - 3$$

Hence we obtain

$$C_{\Omega_0}(B_n) \leq 3(2^{n-k} + 2^{2^k}) + 6 \cdot 2^{2^{k-1}}.$$

For $k := \lfloor \log n \rfloor - 1$, we obtain

$$C_{\Omega_0}(B_n) \leq 3(2^{n-(\lfloor \log n \rfloor - 1)} + 2^{2^{\lfloor \log n \rfloor - 1}}) + 6 \cdot 2^{2^{\lfloor \log n \rfloor - 2}}$$

$$\leq 12 \cdot 2^n \cdot n^{-1} + o(2^n n^{-1}).$$

So it was easy to improve the gap between the obvious upper bound of $n \cdot 2^n + n$ and the lower bound $n^{-1} \cdot 2^n$ by the factor $12^{-1} \cdot n^2$. To eliminate the factor 12 much more work has to be done.

The idea is to use a clever representation of the functions $f: \{0,1\}^n \to \{0,1\}$ found by Lupanov in 1958.

(k,s) - Lupanov - representation:

Interpret f as a function on two sets $\{x_1, x_2, \ldots, x_k\}$ and $\{x_{k+1}, x_{k+2}, \ldots, x_n\}$ of variables.

Example:

				x_4	0	1	0	1	0	1	0	1	
				x_5	0	0	1	1	0	0	1	1	
				x_6	0	0	0	0	1	1	1	1	
	x_1	x_2	x_3										
2^k rows	0	0	0										A_1
	0	0	1										
	0	1	0										
	0	1	1										A_2
	1	0	0				(v)						
	1	0	1										
	1	1	0										A_3
	1	1	1										

2^{n-k} columns

$a = (x_1, x_2, \dots, x_k)$ $\quad b = (x_{k+1}, x_{k+2}, \dots, x_n)$

- Fix s and separate the 2^k rows into $p := \lceil \frac{2^k}{s} \rceil$ groups $A_1, A_2, \dots, A_p$ such that
  - $A_1, A_2, \dots, A_{p-1}$ consist of s rows and
  - A_p consists of $2^k - (p-1)s =: s'$ rows.

- For $1 \le i \le p$ define the functions
$$f_i : \{0,1\}^n \to \{0,1\}$$
by
$$f_i(x) = \begin{cases} f(x) & \text{if } a \in A_i \\ 0 & \text{otherwise} \end{cases}$$
where $x = a, b$.

Then there holds

$$(*) \quad f(x) = \bigvee_{i=1}^{p} f_i(x).$$

Consider $x = a,b$ with b fixed and $a \in A_i$.
Then $f_i(a,b)$ defines an

$$\begin{cases} s\text{-tupel } v & \text{if } 1 \leq i < p \\ s'\text{-tupel } v & \text{if } i = p. \end{cases}$$

Let

$$B_{i,v} := \{ b \in \{0,1\}^{n-k} \mid v \text{ is the tupel of the values of } f_i \text{ with respect to } A_i \}$$

Let

$$f^c_{i,v}(b) : \{0,1\}^{n-k} \to \{0,1\}$$

be defined by

$$f^c_{i,v}(b) := \begin{cases} 1 & \text{if } b \in B_{i,v} \\ 0 & \text{otherwise.} \end{cases}$$

This means that $f^c_{i,v}$ is the characteristic function of the set $B_{i,v}$.

For i,v with $B_{i,v} \neq \emptyset$, we define the function

$$f^r_{i,v}(a) : \{0,1\}^k \to \{0,1\}$$

by

$$f^{r}_{i,v}(a) = \begin{cases} v_j & \text{if } a \text{ is the } j\text{-th element of } A_i \\ 0 & \text{if } a \text{ is not contained in } A_i \end{cases}$$

Then we obtain for $x = a,b$

$$f_i(x) = \bigvee_{v} f^{r}_{i,v}(a) \wedge f^{c}_{i,v}(b).$$

After the insertion of this representation of $f_i(x)$, $x = a,b$ into $(*)$, we obtain the following (k,s)-Lupanov-representation of the function f.

$$f(x) = \bigvee_{i=1}^{p} \bigvee_{v} f^{r}_{i,v}(a) \wedge f^{c}_{i,v}(b).$$

Goal:
The development of an efficient realization of the (k,s)-Lupanov-representation of an arbitrary function $f: \{0,1\}^n \to \{0,1\}$.

We shall use an efficient decoder-network of the decoder-function $f^{(t)}_{decode}: \{0,1\}^t \to \{0,1\}^{2^t}$ which is defined by

$$f^{(t)}_{decode}(x_{t-1}, x_{t-2}, \ldots, x_1, x_0) = (y_{2^t-1}, y_{2^t-2}, \ldots, y_1, y_0)$$

where

$$y_i := \begin{cases} 1 & \text{if } i = \sum_{j=0}^{t-1} x_j \cdot 2^j \\ 0 & \text{otherwise.} \end{cases}$$

This means that $y_i = 1$ iff $x_{t-1} x_{t-2} \cdots x_0$ is the binary representation of the number i.

We shall show that

$$C_{\Omega_0}(f^{(t)}_{decode}) \leq 2^t + (2t-2)\, 2^{\frac{t}{2}}.$$

Consider $f^{\Gamma}_{i,v}$ for v fixed. Let

$$v = \begin{cases} (v_1, v_2, \ldots, v_s) & \text{if } i < p \\ (v_1, v_2, \ldots, v_{s'}) & \text{if } i = p \end{cases}$$

and

$$A_i = \begin{cases} (a_{i_1}, a_{i_2}, \ldots, a_{i_s}) & \text{if } i < p \\ (a_{i_1}, a_{i_2}, \ldots, a_{i_{s'}}) & \text{if } i = p \end{cases}$$

Using the decoder-function $f^{(t)}_{decode}$, we shall construct an efficient realization of these functions

For $a_{ij} \in A_i$, let $[a_{ij}]$ denote the unique integer such that a_{ij} is the binary representation of $[a_{ij}]$.

Then

$$f^{\Gamma}_{i,v}(a) = \bigvee_{j=1}^{|A_i|} y_{[a_{ij}]} \wedge v_j$$

where $f^{(t)}_{decode}(a) = (y_{2^t-1}, y_{2^t-2}, \ldots, y_1, y_0)$.

Note that $y_{[a_{ij}]} = 1$ iff $a = a_{ij}$.

Assume that we have a decoder-network for the decoder function $f^{(k)}_{decode}$ which uses

$$\leq 2^k + (2k-2)\, 2^{\frac{k}{2}}$$

gates.

Then we can realize $f^{\Gamma}_{i,v}$ i, v fixed using at most

s additional $\wedge$-gates and
$s-1$ additional $\vee$-gates

In total we need for fixed v

$$\leq \quad 2ps \quad \leq 2^{k+1}$$

gates.

At most 2^s tupels v are possible. Hence, all these functions can be realized using

$$\leq 2^k + (2k-2)\, 2^{k/2} + 2^s 2^{k+1} = O(2^{k+s})$$

gates.

For the realization of the functions $f^{C}_{i,v}$, we use the decoder function $f^{(n-k)}_{decode}$.

Assume that we have a decoder-network for $f^{(n-k)}_{decode}$ which uses

$$\leq 2^{(n-k)} + (2(n-k)-2)\, 2^{\frac{(n-k)}{2}}$$ gates.

Let

$$B_{i,v} = \{ b_{i_1}, b_{i_2}, \dots, b_{i_\ell} \}.$$

Then

$$f^c_{i,v}(b) = \bigvee_{j=1}^{\ell} y[b_{i_j}]$$

where

$$f^{(n-k)}_{\text{decode}}(b) = (y_{2^{n-k}-1}, y_{2^{n-k}-2}, \dots, y_1, y_0).$$

Note that for fixed i, the sets $B_{i,v}$ are pairwise disjoint.

Hence, for fixed i, all functions $f^c_{i,v}$ can be realized using at most 2^{n-k} $\vee$-gates.

$\Rightarrow$

All these functions can be realized using

$$\leq p \cdot 2^{n-k} + 2^{n-k} + (2(n-k)-2)\, 2^{\frac{n-k}{2}}$$

$$= p \cdot 2^{n-k} + O(2^{n-k})$$

gates.

Using the realizations of the functions $f^r_{i,v}$ and $f^c_{i,v}$, we can construct a network for the computation of f using

- for all i,v one additional $\wedge$-gate, i.e., in total $\leq p \cdot 2^s$ additional $\wedge$-gates

and

- at most $p \cdot 2^s$ additional $\vee$-gates.

Let $C_{k,s}(f)$ denote the total number of gates needed to realize the (k,s)-Lupanov-representation of f.

Altogether, we have proved

$$C_{k,s}(f) \leq O(2^{k+s}) + p \cdot 2^{n-k} + O(2^{n-k}) + O(p 2^s).$$

Since $p = \lceil \frac{2^k}{s} \rceil$, we obtain

$$C_{k,s}(f) \leq O(2^{k+s}) + \frac{2^n}{s} + O(2^{n-k}) + O\left(\frac{2^{k+s}}{s}\right)$$

If we choose

$$k = \lceil 3 \log n \rceil \quad \text{and} \quad s = \lceil n - 5 \log n \rceil$$

then we obtain

$$C_{k,s}(f) \leq O\left(\frac{2^n}{n^2}\right) + \frac{2^n}{n - 5\log n} + O\left(\frac{2^n}{n^3}\right) + O\left(\frac{2^n}{n^3}\right)$$

To complete the construction, we have to describe the decoder-network. First, we need a notation.

Let $a = (a_1, a_2, \ldots, a_t) \in \{0,1\}^t$. The <u>minterm</u> $m_a(x)$ <u>with respect to a</u> is defined by

$$m_a(x) := x_1^{a_1} \wedge x_2^{a_2} \wedge \cdots \wedge x_t^{a_t}.$$

We can realize the decoder-function $f^{(t)}_{\text{decode}}$ directly by using for each y_i, $0 \leq i \leq 2^t - 1$ its corresponding minterm.

For each minterm, we need

- $t-1$ $\wedge$-gates and $\leq t$ negations

Hence, such a realization would use

$$\sim (2t-1) \cdot 2^t$$

gates.

Note that a minterm with respect to t variables consists of the conjunction of

- a minterm with respect to the first $\lfloor \frac{t}{2} \rfloor$ variables

and

- a minterm with respect to the furthes $\lceil \frac{t}{2} \rceil$ variables.

$\rightsquigarrow$

<u>Idea</u>: (Assume that t is even)

Realize $f^{(t)}_{decode}$ by the conjunction of

- all minterms generated by a network for $f^{(\frac{t}{2})}_{decode}$ with respect to the variables $x_{\frac{t}{2}-1}, x_{\frac{t}{2}-2}, \dots, x_1, x_0$

with

- all minterms generated by a network for $f^{(\frac{t}{2})}_{decode}$ with respect to the variables $x_{t-1}, x_{t-2}, \dots, x_{\frac{t}{2}}$.

$\Rightarrow$

$$C_{\Omega_0}(f^{(t)}_{decode}) \leq 2 \cdot C_{\Omega_0}(f^{(\frac{t}{2})}_{decode}) + 2^n.$$

If we use the direct realization for the two functions $f^{(\frac{t}{2})}_{decode}$ then we obtain.

$$C_{\Omega_0}(f^{(t)}_{decode}) \leq 2^t + (2t-2)\, 2^{\frac{t}{2}}$$

This finishs the constraction.

Altogether, we have proved the following theorem.

Theorem 1.2

For all $\varepsilon > 0$ there is $N_0 > 1$ such that for all $n \geq N_0$ for each function $f: \{0,1\}^n \to \{0,1\}$

$$C_{\Omega_0}(f) \leq (1+\varepsilon)\frac{2^n}{n}.$$

Exercise

For the lower bound proof (Theorem 1.1) we have chosen the base Ω_0. Which lower bound could you prove if you would choose the base B_2 instead of Ω_0?

2. Lower bounds on the network complexity of Boolean functions

References

- Claus-Peter Schnorr, Zwei lineare untere Schranken für die Komplexität Boolescher Funktionen, Computing 13 (1974), 155 - 171.
- Wolfgang Paul, A 2.5n-lower bound on the combinational complexity of boolean functions, SIAM J. Comput. 6 (1977), 427 - 443.
- Larry Stockmeyer, On the combinational complexity of certain symmetric Boolean functions, Math. Systems Theory 10 (1977), 323 - 336.
- Norbert Blum, A Boolean function requiring 3n network size, TCS 28 (1984), 337 - 345.
- Uri Zwick, A 4n lower on the combinational complexity of certain symmetric Boolean functions over the basis of unate dyadic Boolean functions, SIAM J. Comput. 20 (1991), 499 - 505.

Although the network complexity of almost all Boolean functions is exponential, no nonlinear lower bound of an explicit defined Boolean function in NP are known. We shall present some of the few known linear lower bound

proofs. First, we shall consider the base B_2.

A function $f \in B_n$ is called <u>degenerated</u> if f does not depend on all variables $x_1, x_2, \ldots, x_n$. This means that there is a variable x_i such that the subfunctions of f after setting $x_i := 0$ and $x_i := 1$ are equal.

<u>Theorem 2.1</u>

$C_{B_2}(f) \geq n-1$ for all nondegenerated functions in B_n.

<u>Proof</u>:

Let β be an optimal B_2-network for f.

<u>Observation</u>

- The outdegree of each input node x_i in β is at least one. (Otherwise, f would not depend on x_i).
- Each gate which is not the output node has outdegree at least one. (Otherwise, we could eliminate this gate without changing the function computed at the output node.)

$\Rightarrow$

Number of edges $\geq n + c - 1$,

where c is the number of gates of indegree two in β.

Each edge has a gate of indegree two as end node.

$\Rightarrow$

$$\text{Number of edges} = 2 \cdot c$$

$\Rightarrow$

$$2c \geq n + c - 1$$

$$\Leftrightarrow \quad c \geq n - 1.$$

The proof of Theorem 2.1 is only based on the graph structure of the network β. A better bound cannot be obtained in this way since there are nondegenerated functions in B_n which have network size $n-1$ (e.g., $f: \{0,1\}^n \to \{0,1\}$ with $f(x_1, x_2, \ldots, x_n) = x_1 \wedge x_2 \wedge \ldots \wedge x_n$).

$\Rightarrow$

We need further techniques to prove larger lower bounds.

All proofs of larger lower bounds use the so-called gate-elimination method. The gate-elimination method uses induction. By an assignment of some variables with values from $\{0,1\}$, a specific number of gates are eliminated in each step and the resulting function is of the same type as the function before the assignment.

We shall demonstrate the gate-elimination method proving some lower bounds. First, we need a definition.

A Boolean function $f \in B_n$ belongs to the class $Q^n_{2,3}$ if for all different $i, j \in \{1, 2, \ldots, n\}$, the following is fulfilled.

i) After the replacement of x_i and x_j by constants, we obtain at least three different subfunctions.

ii) If $n \geq 4$ then there is a constant c_i such that after the replacement of x_i by c_i, we obtain a subfunction in $Q^{n-1}_{2,3}$.

Schnorr has defined the class $Q^n_{2,3}$ in such a way that a lower bound $2n-3$ for each function in this class can be proven easily.

Theorem 2.2

For all $f \in Q^n_{2,3}$ there holds $C_{B_2}(f) \geq 2n-3$.

Proof:

Observe that each function $f \in Q^n_{2,3}$ is node-generated. If f would not depend on x_i then f could have at most two different subfunctions with respect to x_i and any $x_j \neq x_i$.

Let β be an optimal B_2-network for f.

Consider a gate v in β such that both direct predecessors of v are input nodes. Obviously, such a gate exists.

Optimality of β $\Rightarrow$
Both input nodes correspond to different variables x_i and x_j.

If both input nodes have outdegree one, then f depends on x_i and x_j only via the gate v. v can only compute 0 or 1.

$\Rightarrow$

f can have at most two different subfunctions with respect to x_i and x_j. This contradicts the definition of $Q^n_{2,3}$.

$\Rightarrow$

At least one of the two input nodes has outdegree at least two.

W.l.o.g. we can assume that the input node with label x_i has outdegree ≥ 2.

We distinguish two cases.

Case 1: $n = 3$

Then at least four edges leave the input nodes. Analogous to the proof of Theorem 2.1, we can show the existence of 3 gates in β.

Case 2: $n \geq 4$

Assume that the assertion is proved for $\ell < n$.
Replace x_i by c_i.

$\Rightarrow$

- At least two successors of the input node x_i can be eliminated.
- The resulting network β' computes a function in $Q_{2,3}^{n-1}$.

$\Rightarrow$

assumption

β' contains at least $2(n-1)-3$ gates.

$\Rightarrow$

β contains at least $2n-3$ gates.

■

Example:

Counting-function:

$$C_{k,m}^{n} : \{0,1\}^n \longrightarrow \{0,1\}$$

where

$$C_{k,m}^{n}(x_1, x_2, \ldots, x_n) = \begin{cases} 1 & \text{if } \sum_{i=1}^{n} x_i = k \bmod m \\ 0 & \text{otherwise} \end{cases}$$

Let us consider $C_{k,3}^{n}$.

After the replacement of x_i and x_j by constants, we obtain the different functions

$$\begin{cases} C_{k,3}^{n-2} & \text{if } x_i + x_j = 0 \\ C_{k-1,3}^{n-2} & \text{if } x_i + x_j = 1 \\ C_{k-2,3}^{n-2} & \text{if } x_i + x_j = 2 \end{cases}$$

For $n \geq 4$, we define $c_i := 0$ for all i.
After the replacement of x_i by c_i, we obtain the subfunction $C_{k,3}^{n-1}$ which is in $Q_{2,3}^{n-1}$.

The so-called storage access function $SA_n \in B_{n+k}$ where $n = 2^k$ is defined on a k-bit number $a = a_1 a_2 \ldots a_k$ and n variables $x_1, x_2, \ldots, x_n$ by

$$SA_n(a, x) = x_{(a)} \quad \text{where}$$

(a) is the binary number represented by $a+1$

Wolfgang Paul has proved a $2n-2$ lower bound for the network complexity of SA_n.

Theorem 2.3

$$C_{B_2}(SA_n) \geq 2n - 2.$$

Proof:
Our goal is to give an inductive proof. But the replacement of x variables does not lead to

storage access functions. Hence, we define a larger class of functions.

Let $s \in \{1,2,\dots,n\}$. F_s contains all functions $f \in B_{n+2}$ such that there exists $S \subseteq \{1,2,\dots,n\}$, $|S| = s$ such that for all a with $(a) \in S$

$$f(a, x_1, x_2, \dots, x_n) = x_{(a)}.$$

Since $SA_n \in F_n$, it suffices to prove a $2s-2$ lower bound for all functions in F_s.

$s = 1$: trivial since $2s-2 = 0$.

Let $s \geq 2$. Assume that the lower bound holds for $\ell < s$; i.e., $C_{\Omega_2}(f) \geq 2\ell - 2$ for all $f \in F_\ell$.

Consider any function $f \in F_s$. Let β be an optimal B_2-network for f. We prove that we obtain a network for a subfunction in F_{s-1} after the elimination of at least two gates.
There cases are possible:

Case 1: $\exists i \in S$: input node x_i has outdegree ≥ 2.

Setting x_i to a constant eliminates at least two gates. Replace S by $S \setminus \{i\}$. The resulting network computes a function $f' \in F_{s-1}$. By assumption $C_{\Omega_2}(f') \geq 2(s-1) - 2$.

$$\Rightarrow C_{B_2}(f) \geq 2s - 2.$$

Case 2: $\exists i \in S$: input node x_i has outdegree one and the edge from x_i goes into an $\wedge$-type gate v; i.e., $res_B(v) = (x_i^b \wedge g^c)^d$ with $b, c, d \in \{0,1\}$ and g is a function of the other input variables.

If we replace x_i by $\bar{b}$ then the output of v is replaced by the constant 0^d. Furthermore, the resulting network computes a function in F_{s-1}.

$\Rightarrow$

v cannot be the output gate.

At least the gate v and its successor have been eliminated. As in Case 1, we obtain

$$C_{B_2}(f) \geq 2s - 2.$$

Case 3: $\exists i \in S$: input node x_i has outdegree one and the edge from x_i goes into a $\oplus$-type gate v.

Then $res_B(v) = x_i \oplus g \oplus b$ for some function g of the other variables and $b \in \{0,1\}$.

Furthermore, v cannot be the output gate. To see this, consider $j \in S \setminus \{i\}$ and $\sigma(a) = j$. The network has to compute x_j independently from the value of x_i. A change of the value of x_i changes the result of the gate v.

The value of x_i has no influence on the output for all $(a) \in S \setminus \{i\}$.

$\Rightarrow$

After replacing x_i by an arbitrary function, we obtain a network for a function in F_{S-1}.

$\rightsquigarrow$

Choose the function g to replace x_i. Then the gate v computes the constant b.

$\Rightarrow$

v and its successors are eliminated.

$\Rightarrow$

At least two gates are eliminated.

As in Case 1, we obtain

$$C_{B_2}(f) \geq 2s-2.$$

□

Wolfgang Paul has also proved a $2.5n$ lower bound for the B_2-complexity for a function

$$f: \{0,1\}^{n+2\log n+1} \longrightarrow \{0,1\},$$

defined in the following way:

Let

$$\sigma_1 = a_1, a_2, \dots, a_{\log n}$$

$$\sigma_2 = a_{\log n+1}, \dots, a_{2\log n}$$

and

$$f(a_1, a_2, \ldots, a_{2\log n}, q, x_0, x_1, \ldots, x_{n-1})$$

$$= \begin{cases} x_{(\sigma_1)} \wedge x_{(\sigma_2)} & \text{if } q = 1 \\ x_{(\sigma_1)} \oplus x_{(\sigma_2)} & \text{if } q = 0. \end{cases}$$

Larry Stockmeyer has applied the ideas of Paul to prove a $2.5n$ lower bound for several fundamental symmetric functions.

A function $f \in B_{n,m}$ is <u>symmetric</u> if

$$f(x_1, x_2, \ldots, x_n) = f(x_{\pi(1)}, x_{\pi(2)}, x_{\pi(3)}, \ldots, x_{\pi(n)})$$

for all permutations π of $1, 2, \ldots, n$.

We have considered the function

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

where

$$f(a_1, a_2, \ldots, a_{3\log n}, r, x_1, x_2, \ldots, x_n)$$

$$= x_{(\sigma_1)}^{r} \wedge (x_{(\sigma_2)} \oplus x_{(\sigma_3)})$$

with

$$\sigma_3 = a_{2\log n+1}, \ldots, a_{3\log n}.$$

We have extended the method of Paul and proved that $C_{B_2}(f) \geq 3n - 3$.

Note that the function f is similar to the function of Paul.

- If $r = 1$ and $x_{(\sigma_1)} = 1$ then

$$f(\sigma_1, \sigma_2, \sigma_3, r, x_1, x_2, \ldots, x_n) = x_{(\sigma_2)} \oplus x_{(\sigma_3)}.$$

- If $r = 1$ and $x_{(\sigma_3)} = 0$ then

$$f(\sigma_1, \sigma_2, \sigma_3, r, x_1, x_2, \ldots, x_n) = x_{(\sigma_1)} \wedge x_{(\sigma_2)}.$$

We shall present the proof of this lower bound.

Throughout the proof, we shall use the following lemma

Lemma 2.1

Let β be a network computing $f \in B_n$. Let $v \in \beta$ be an $\wedge$-type gate or a $\oplus$-type gate. If one input function of v is constant then we can eliminate gate v and the reduced network still computes f.

Proof:

exercise

∎

Let $U \subset V_n$ and $\alpha : U \to \{0,1\}$ be a mapping. Frequently, we shall consider the restriction f_α of $f \in B_n$ under the assignment α. More precisely, f_α is defined by

$f_\alpha(x_1, x_2, \ldots, x_n) = f(y_1, y_2, \ldots, y_n)$ 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. $(v \Rightarrow v)$ denotes a path from node v to node u. $(v \Rightarrow u[$ 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:

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

$$f(\sigma_1, \sigma_2, \sigma_3, r, x_1, \ldots, x_n) = x_{(\sigma_1)}^{r} \wedge (x_{(\sigma_2)} \oplus x_{(\sigma_3)})]$$

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 v in β with $op(v) = x_i$.

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 B $\Rightarrow$ 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 B 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 $\#\,suc(u_r) = 1$.

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

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

$\Rightarrow$

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

In both subcases, an application of the induction hypothesis gets

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

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

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

Then there exists nodes $u_1, u_2, \dots, u_r, v_1, v_2, \dots, v_\ell \in B$ with

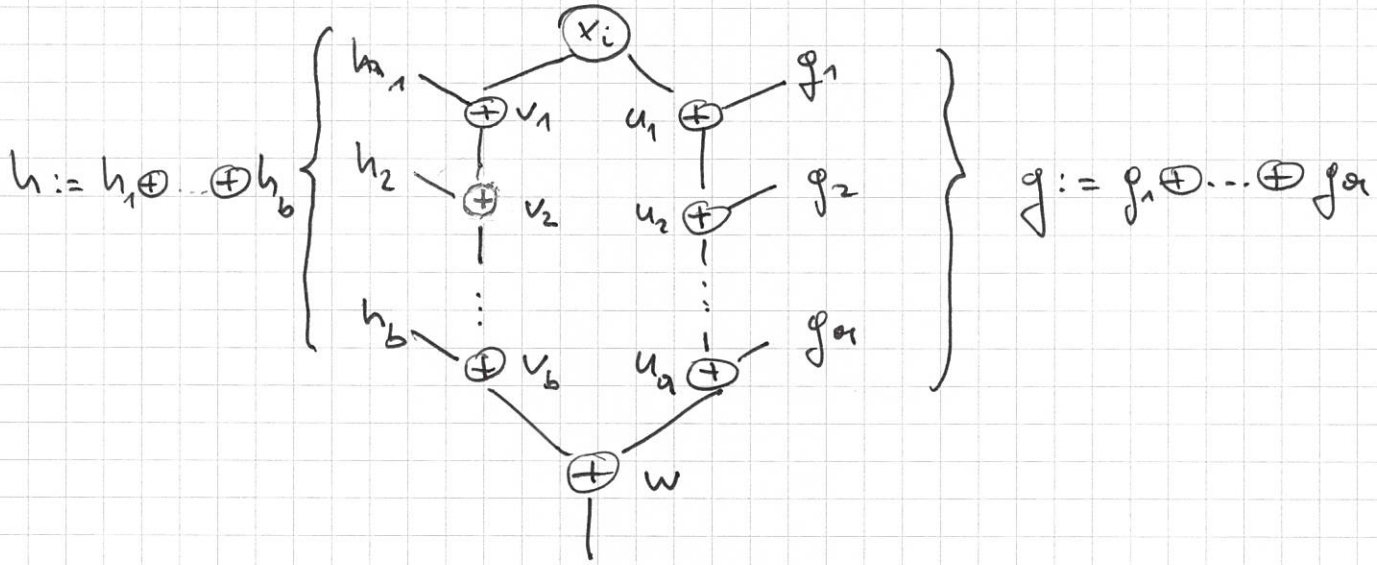
1) $u_1 \in suc(x_i)$,
2) u_j is a $\oplus$-type gate for $1 \leq j \leq r$
3) $u_{j+1} \in suc(u_j)$ and $\#\,suc(u_j) = 1$ for $1 \leq j < r$
4) $\#\,suc(u_r) > 1$ or $w \in suc(u_r)$ is an $\wedge$-type gate

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:



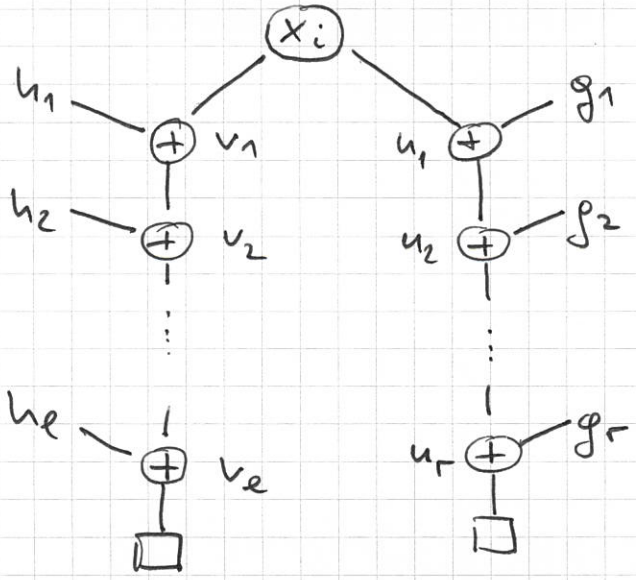
Then

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

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

□

Therefore, we have the following situation.



β acyclic $\Rightarrow$

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

or

$h_1, h_2, \dots, h_\ell$ 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 $g_1, g_2, \dots, g_r$ 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 $f(\sigma_1, \sigma_2, \sigma_3, \tau, x_1, x_2, \dots, x_n) = c \oplus x_j$ for

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

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 $Suc(G_i)$. 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 $\#Suc(G_i) = 1$.

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 $res(t) = f$. 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) $u_1, u_2 \in Suc(w)$,

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

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

□

Lemma 2.5

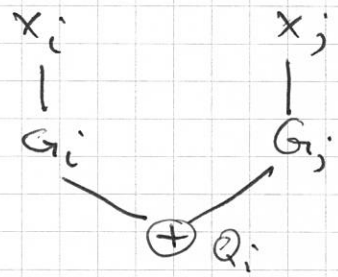
$\forall i,j \in S$ holds: $i \neq j \Rightarrow Q_i \neq Q_j$.

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.

Here, 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.

□

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 $\Rightarrow$

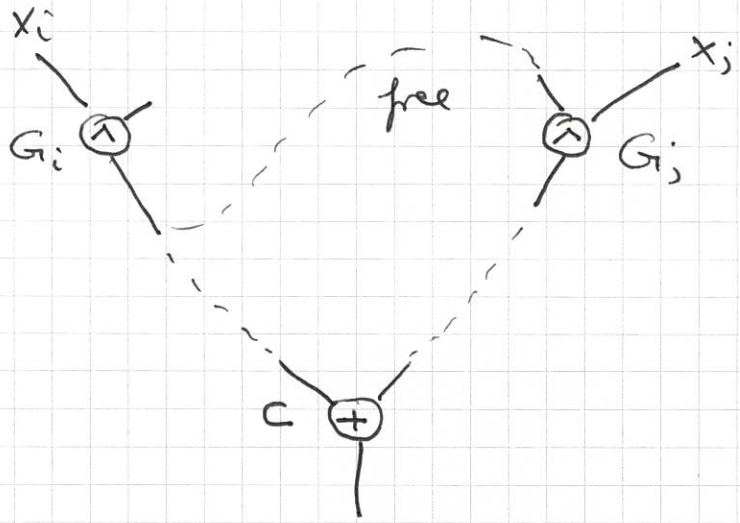
$\exists\, i, j \in S$, $i \neq j$ with the following property:

- 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) $\mathrm{res}(G_v)_\alpha$ 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_e at $a \in \{0,1\}$ such that

$\mathrm{res}(G_e)_{\alpha, x_e := a}$ is constant.

$\Rightarrow$

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

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

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

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

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 $\ell, v \in S \setminus \{j\}$, $v \neq \ell$ 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

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

Then

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

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 nonlinear lower bound for the network complexity of a Boolean function it suffices to prove this with respect to the base Ω_0. Such a nonlinear lower bound would imply a nonlinear 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 (can be obtained from the home-page of Ran Raz)

a $5n$ lower bound is claimed. The proof ideas looks 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 an explicit Boolean function are known.

We shall consider monotone Boolean networks first.