

The complexity of Boolean functions

References

- Ingo Wegener, The Complexity of Boolean Functions, Teubner - Wiley, 1987.
- Poul E. Dunne, The Complexity of Boolean Networks, Academic Press, 1988.
- 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, 758 - 804, 1990.
- Norbert Blum, On Negations in Boolean Networks, LNCS 5760, 18 - 28, 2008.
- Stasys Jukna, Boolean Functions Complexity: Advances and Frontiers, Springer, 2012.

0. Motivation

(2)

Is $P = NP$? This means, is deterministic polynomial time equal nondeterministic polynomial time? The P versus NP - question is the most famous open problem in Computer Science.

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

Theorem 0.1

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

Proof:

Not hard. See for example

John E. Savage, Models of Computation:
Exploring the Power of Computing,
Addison-Wesley, 1998, pp. 124 - 127.

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

Theorem 0.1 $\Rightarrow$

The proof of a super-polynomial lower bound for the network complexity of the characteristic function of a language $L \in NP$ implies $P \neq NP$.

Besides the famous P versus NP problem there are other major problems of theoretical computer science which could be attacked using Boolean complexity:

- Can we multiply two integers in linear time or can we prove an $\Omega(n \log n)$ lower bound for the network complexity of the multiplication of two n -bit numbers?
- What is the network complexity of the n -th degree convolution or of the Boolean matrix multiplication?
- $\vdots$

Main topic of the lecture:

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

1. Preliminaries and asymptotic results

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

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

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

Variables and negated variables are called literals.

A function $m: \{0,1\}^n \rightarrow \{0,1\}$ which is the conjunction of some literals is called a monomial.

If we delete some literals from a monomial m then we obtain a submonomial m' of m and write $m' \subseteq m$. The empty monomial ε is the constant function 1. The disjunction of monomials is a formula in disjunctive normal form (DNF). The disjunction of some literals is called a clause. If we delete some literals from a clause d then we obtain a subclause of d . The empty clause is the constant function 0. The conjunction of clauses is a formula in conjunctive normal form (CNF).

A monomial m is called an implicant of the function f if for all $a \in \{0,1\}^n$, $m(a) = 1$ implies $f(a) = 1$. An implicant m is a prime implicant of f if no proper submonomial of m is an implicant of f .

A clause d is called an f -clause if for all $a \in \{0,1\}^n$, $d(a) = 0$ implies $f(a) = 0$. A prime clause d of f is an f -clause where no proper subclause of d is an f -clause.

$IM(f)$ denotes the set of all implicants of f .

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

$PCL(f) \subseteq CL(f)$ is the set of all prime clauses of f .

Let $a := (a_1, a_2, \dots, a_n)$, $b := (b_1, b_2, \dots, b_n) \in \{0,1\}^n$. We write $a \leq b$ iff $a_i \leq b_i$ for $1 \leq i \leq n$.

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

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

B_2 is the set of basic operations. For $\Omega \subseteq B_2$, an Ω -network (or Ω -circuit) β is a directed acyclic graph such that each node has indegree at most two. The nodes g with indegree zero are input nodes and are labelled with $op(g) \in V_n$.

The nodes g with indegree larger than zero are the gates of β . Each gate g is labelled with an operator $op(g) \in \Omega$ where the indegree of g is equal the number of operands of $op(g)$. A node with

outdegree zero is an output node.

For a node g in β let

$$\text{Suc}(g) := \{h \mid g \rightarrow h \text{ is an edge in } \beta\}$$

and

$$\text{pred}(g) := \{h \mid h \rightarrow g \text{ is an edge in } \beta\}$$

be the sets of direct successors and direct predecessors of g .

With each node g , we associate a function

$\text{res}_\beta(g) : \{0,1\}^n \rightarrow \{0,1\}$ which is defined by

$$\text{res}_\beta(g) := \begin{cases} \text{op}(g) & g \text{ is an input node,} \\ \neg \text{res}_\beta(h_1) & \text{op}(g) = \neg, \text{pred}(g) = \{h_1\} \\ \text{res}_\beta(h_1) \text{ op}(g) \text{ res}_\beta(h_2) & \text{otherwise,} \\ & \text{where pred}(g) = \{h_1, h_2\}. \end{cases}$$

The functions $\text{res}_\beta(g)$ with g is a node in β are computed by β . Let $G \in B_n$. The minimum number of gates in an Ω -network which computes G is the Ω -complexity $C_\Omega(G)$ of G . Usually, not-gates are not counted.

For $f \in B_n$ and $a \in \{0,1\}$ let

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

⑦

$f \in B_2$ is $\wedge$ -type if $\exists a, b, c \in \{0, 1\}$ such that

$$f(x, y) = (x^a \wedge y^b)^c$$

$f \in B_2$ is $\oplus$ -type if $\exists a \in \{0, 1\}$ such that

$$f(x, y) = (x \oplus y)^a$$

Lemma 1.1

The functions $f \in B_2$ can be classified in the following way. There are

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

Proof:

exercise



Exercise:

Show that $|B_n| = 2^{2^n}$ and $|B_{n,m}| = 2^{m2^n}$.

Each Σ -network can also be viewed as a straight line program $g_1, g_2, \dots, g_t$, $t \geq n$ of 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 $1 \leq j < i$ or $g_i = g_{i_1} \text{ op } (g_{i_2})$ where $1 \leq i_1, i_2 < i$ and $\text{op}(g_i) \in \Omega \setminus \{\neg\}$.

Goal:

The development of lower and upper bounds for the Ω_0 -complexity, $\Omega_0 := \{\wedge, \vee, \neg\}$ of not explicitly defined functions in $\mathcal{B}_n$.

Let

$$\Phi(n, t) := |\{f \in \mathcal{B}_n \mid C_{\Omega_0}(f) \leq t\}|.$$

First we will estimate an upper bound for $\Phi(n, t)$.

Lemma 1.2

$$\Phi(n, t) \leq t^{(t+1)} \cdot 9^t \cdot e^{2n}$$

Proof:

Obviously, $\Phi(n, t)$ is upper bounded by the number of distinct Ω_0 -networks with $\leq t$ gates and n input nodes.

Goal:

Derivation of an upper bound for the number of distinct Ω_0 -networks with exactly $t' \leq t$ gates and n input nodes.

For each gate there are

- $|\Omega_0| = 3$ possibilities to choose its operator and

(9)

• $\leq t' - 1 + n$ possibilities to choose each of its direct 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 Ω_0 -networks with t' gates and n input nodes is upper bounded by

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

$\Rightarrow$

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

Sterling's formula $\Rightarrow$

$$t! \geq t^t \cdot e^{-t} > \left(\frac{t}{3}\right)^t$$

where $e = 2,71828 \dots$ is the Euler number.

Furthermore, we know that $1 + x \leq e^x$.

Hence

$$\begin{aligned} \phi(n, t) &\leq \frac{t \cdot 3^t (t + n)^{2t}}{t^t} \\ &= t \cdot 3^t t^t \left(1 + \frac{n}{t}\right)^{2t} \\ &\leq t^{(t+1)} 3^t e^{2n} \end{aligned}$$



To compute all functions in $\mathcal{B}_n$ by an Ω_0 -net = work with at most t gates, we need

$$\phi(t, n) \geq |\mathcal{B}_n| = 2^{2^n}$$

If we choose t such that $\phi(t, n) \geq |\mathcal{B}_n|$, Lemma 1.2 implies

$$t^t \cdot t \cdot 9^t \cdot e^{2n} \geq 2^{2^n}$$

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

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

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

$$\begin{aligned} t \cdot \log t + 4t + 4n &\leq \frac{2^n}{n} (n - \log n + 4) + 4n \\ &= 2^n - \frac{\log n - 4}{n} \cdot 2^n + 4n. \end{aligned}$$

But for $n > 32$, we obtain

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

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

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

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

Let $\mathcal{B}_n^* \subset \mathcal{B}_n$ be the set of those $2^{a(n)}$ functions in $\mathcal{B}_n$ 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:

Prove for n large enough that B_n^* contains a function which needs 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 proved the following theorem.

Theorem 1.1

For sufficiently large n , at least

$(1 - 2^{-2^n n^{-1} \log \log n}) \cdot |B_n|$
of the 2^{2^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 be computed by an $\mathcal{D}$ -network which has at most $n \cdot 2^n + n$ gates.

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

a) The decoding network

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

$$(*) \quad f = (\bar{x}_n \wedge f_0) \vee (x_n \wedge f_1).$$

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 $C_{\Omega_0}(\mathcal{B}_n)$ denote the minimum number of gates without negations of input nodes which suffices to realize any function in $\mathcal{B}_n$. Then we obtain for $n \geq 2$

$$C_{\Omega_0}(\mathcal{B}_n) \leq 2 C_{\Omega_0}(\mathcal{B}_{n-1}) + 3.$$

Note that

$$C_{\Omega_0}(\mathcal{B}_2) = 3.$$

Exercise

Show that $C_{\Omega_0}(\mathcal{B}_2) = 3$.

$\Rightarrow$

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

Note that

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

Hence we obtain

$$\begin{aligned} C_{\Omega_0}(B_n) &\leq 3 \cdot 2^{n-2} + 3 \cdot 2^{n-2} - 3 \\ &= 3 \cdot 2^{n-1} - 3. \end{aligned}$$

This improves the obvious upper bound by the factor n .

b) The improvement of the decoding network.

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 to use them if needed.

More generally, let $C_{\Omega_0}^*(B_k)$ be the minimum number of gates used for the computation of all functions in B_k where the negations of input nodes are not counted. Because of (*), we obtain

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

Exercise

Show that $C_{\Omega_0}^*(B_2) \leq 12$.

$$\leq 3 \cdot 2^{2^k} + 3 \left(\sum_{i=2}^{k-1} |B_i| \right)$$

$$= 3 \cdot 2^{2^k} + 3 \left(\sum_{i=2}^{k-1} 2^{2^i} \right)$$

$$\leq 3 \cdot 2^{2^k} + 3 \left(\left(1 + \frac{1}{2} + \frac{1}{4} + \dots \right) 2^{2^{k-1}} \right)$$

$$\leq 3 \cdot 2^{2^k} + 6 \cdot 2^{2^{k-1}}$$

The computation of the function f can be realized in the following way.

- (1) Choose the appropriate k .
- (2) Compute all functions in B_k .
- (3) Using B_k , apply the improved 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

$$\leq 3 \cdot 2^{n-k} - 1.$$

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

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

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

- 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 $2^k - (p-1) \cdot s =: s'$ rows.
- For $1 \leq i \leq p$ define the functions $f_i: \{0,1\}^n \rightarrow \{0,1\}$ for $x = a, b$ by

$$f_i(x) := \begin{cases} f(x) & \text{if } a \in A_i \\ 0 & \text{if } a \notin A_i. \end{cases}$$

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 $f_i(a, b)$ define an

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

Let

$$B_{i,v} := \{ b \in \{0,1\}^{n-k} \mid v \text{ is the tuple of the values of } f_i \text{ w.r.t. } A_i \}.$$

Let

$$f_{i,v}^c(b) : \{0,1\}^{n-k} \rightarrow \{0,1\}$$

be defined by

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

This means that $f_{i,v}^c(b)$ is the characteristic function of the set $B_{i,v}$.

For i, v with $B_{i,v} \neq \emptyset$, we define the function $f_{i,v}^r(a) : \{0,1\}^k \rightarrow \{0,1\}$

by

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

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

$$f_{i,v}^r(a) : \{0,1\}^k \rightarrow \{0,1\}$$

by $f_{i,v}^r(a) := 0$.

Then we obtain for $x = a, b$

$$f_i(x) = \bigvee_v f_{i,v}^r(a) \wedge f_{i,v}^c(b)$$

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

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

Goal:

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

We will use an efficient decoder-network of the decoder function $f_{\text{decode}}^{(t)}: \{0, 1\}^t \rightarrow \{0, 1\}^{2^t}$ which is defined by

$$f_{\text{decode}}^{(t)}(x_{t-1}, x_{t-2}, \dots, x_1, x_0) = (y_{2^t-1}, y_{2^t-2}, \dots, y_1, y_0)$$

where

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

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

We will show that

$$C_{\Omega_0}(f_{\text{decode}}^{(t)}) \leq 2^t + (2t-2) 2^{\frac{t}{2}}$$

Consider $f_{i,v}^r$ for v fixed. Let

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

and

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

Using the decoder-function $f_{\text{decode}}^{(k)}$, we 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

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

where $f_{\text{decode}}^{(k)}(a) = (y_{2^k-1}, y_{2^k-2}, \dots, y_1, y_0)$.

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

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

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

gates.

Then for fixed i, v , we can realize $f_{i,v}^r$ using at most

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

For fixed v , we need in total

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

additional gates.

At most 2^s tuples 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 function $f_{i,v}^c$, we use the decoder-function $f_{\text{decode}}^{(n-k)}$. Assume that we have a decoder-network for $f_{\text{decode}}^{(n-k)}$ which uses

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

gates. Let

$$B_{i,v} = \{b_{i,1}, b_{i,2}, \dots, b_{i,2^{n-k}}\}.$$

Then

$$f_{i,v}^c(b) = \bigvee_{j=1}^{2^{n-k}} y[b_{i,j}]$$

where

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

Note that for fixed i , the sets $B_{i,v}$ are pairwise disjoint. Hence, for fixed i , all functions $f_{i,v}^c$ can be realized using at most 2^{n-k} additional v -gates.

$\Rightarrow$

All these functions can be realized using

$$\begin{aligned} &\leq p \cdot 2^{n-k} + 2^{n-k} + (2(n-k) - 2) 2^{\frac{n-k}{2}} \\ &= p \cdot 2^{n-k} + O(2^{n-k}) \end{aligned}$$

gates.

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

- $\forall i, v$ one additional x -gate; i.e., in total $\leq p \cdot 2^s$ additional $\wedge$ -gates and
- at most $p \cdot 2^s$ additional v -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

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

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 $a = (a_1, a_2, \dots, a_t) \in \{0, 1\}^t$. For $x = (x_1, x_2, \dots, x_t)$, the monomial $m_a(x)$ with respect to a is defined by

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

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

For each monomial, we need

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

Hence, such a realization would use

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

gates.

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

- a monomial with respect to the first $\lfloor \frac{t}{2} \rfloor$ variables and
- a monomial with respect to the further $\lceil \frac{t}{2} \rceil$ variables.

$\leadsto$

Idea: (Assume that t is even)

Realize $f_{\text{decode}}^{(t)}$ by the conjunction of

- all monomials generated by a network for $f_{\text{decode}}^{(t/2)}$ with respect to the variables $x_{t/2-1}, x_{t/2-2}, \dots, x_1, x_0$ with
- all monomials generated by a network for $f_{\text{decode}}^{(t/2)}$ with respect to the variables $x_{t-1}, x_{t-2}, \dots, x_{t/2}$.

$\Rightarrow$

$$C_{\Omega_0}(f_{\text{decode}}^{(t)}) \leq 2 \cdot C_{\Omega_0}(f_{\text{decode}}^{(t/2)}) + 2^t.$$

If we use the direct realization for the two functions $f_{\text{decode}}^{(t/2)}$ then we obtain

$$\begin{aligned} C_{\Omega_0}(f_{\text{decode}}^{(+)}) &\leq 2^t + 2 \cdot (2^{t/2} - 1) 2^{t/2} \\ &= 2^t + (2^t - 2) 2^{t/2} \end{aligned}$$

This finishes the construction.

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 $f: \{0,1\}^n \rightarrow \{0,1\}$

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

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 $\mathcal{B}_2$ instead of the base Ω_0 ?
- Give an estimation of N_0 (Theorem 1.2) in dependence of ε .