4. The approximation method

References

- Alexander E. Andreev, On a method for obtaining efficient lower bounds for monotone complexity, Algebra and Logik (1987) 26:1. https://10.1007/BF01978380.
- Alexander A. Razborov, A lower bound on the monotone network complexity of the logical permanent, Math. Notes Acad. Sci. USSR 37 (1985), 485 - 493.
- Noga Alon, Ravi B. Boppana, The monotone circuit complexity of Boolean functions, Combinatorica 7(1987), 1 - 22.
- Ingo Wegener, The Complexity of Boolean Functions, Teubner - Wiley, 1987, 180 - 183.
- Paul E. Dunne, The Complexity of Boolean Networks, Academic Press, 1988, 185 - 213.
- Stasys Jukna, Boolean Function Complexity: Advances and Frontiers, Springer, 2012, 274 - 276.

In 1984, Razborov has introduced the approximation method in Boolean complexity to prove lower bounds for the monotone complexity of certain monotone Boolean functions. Independently and at the same

time, Andreev has introduced a similar approach. We will investigate Razborov's method. Before introducing the approximation method, some general remarks about

- which arguments should be used in a proof and
- how a proof should be presented

are given.

Assume that we want to prove a theorem; for example a lower bound for the monotone complexity of the clique function. To prove such a theorem, some lemmas are proved. Some rules should be observed with respect to such a proof.

The first rule is obvious.

1) Prove only a lemma if the lemma is used in the proof of the theorem.

The second and the third rule say something about the assumptions used to prove the theorem.

2) All assumptions of a lemma have to be used in the proof of the lemma.

We could remove an assumption not used in the proof and still, the proof would prove the modified lemma.

Sometimes, the addition of an assumption could simplify the proof of the lemma considerably and the weaker lemma is strong enough for the proof of the theorem. In such a situation, the third rule applies.

3) If a weaker lemma with a simpler proof could be used in the same way as the strong lemma to prove a theorem, use the weaker lemma.

If we have a proof of a theorem then we have to decide how to present the proof in a paper or in a talk. This should be based on our goals.

First of all,

- the reader should understand the proof.

Moreover,

- the reader should get knowledge about the intuition that lead to the idea of proof.

In addition to this

- the presentation should take into account what is still intended. For example, do we want to try the generalization of the method?

Now we will investigate the presentations of Razborov's approximation method as given by

Razborov, Alon and Boppana, Wegener, Dunne, and Jukna.

The method has been developed to prove super-polynomial lower bounds for the monotone complexity of some Boolean functions like the clique function. To get a method to prove such a lower bound, the usual way is to start with an optimal monotone network which computes the considered Boolean function. The goal is to get an intuition leading to a general idea how to prove the lower bound. Then one looks for a concrete method which implements the idea.

In all presentations of Razborov's approximation method, the way from the monotone network to the concrete method is missing. They start with the concrete method, derive properties and finally, prove the lower bound. The intuition behind the proof is hidden. Because of this, it is hard to get an intuition with respect to the generalization of the method to non-monotone Boolean networks. In the lecture, I shall close this gap.

To get a lower bound for the monotone complexity of a monotone function $f \in B_n$, we start with a monotone network β which computes f. We have no knowledge about the structure of β. In particular, we have no knowledge about the DNF-representations of the functions computed at the nodes of β. 116a)

Let g be any node in a network 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 of β, we can compute these DNF-formulas by applying the properties of the Boolean operations. We call this representation of $res_\beta(g)$ the <u>DNF-representation</u> $DNF_\beta(g)$ of $res_\beta(g)$.

The function $res_\beta(g)$ can be written as a CNF-formula as well; i.e.,

$$res_\beta(g) = \bigwedge_{j=1}^{s} d_j ,$$

where each d_j is a clause. We denote this formula $CNF_\beta(g)$ the <u>CNF-representation</u> of $res_\beta(g)$.

<u>Exercise</u>

Develop an algorithm which computes for each node g of a given monotone network β $DNF_\beta(g)$ and $CNF_\beta(g)$.

Next we shall characterize $DNF_\beta(g_t)$ where g_t is the output node of β. Let $p_1, p_2, \dots, p_r$ be the prime implicants of the function computed by β and let

$$DNF_\beta(g_t) = \bigvee_{j=1}^{t_0} m_j .$$

Then each monomial m_j, $1 \le j \le t_0$ is an implicant

of f. Otherwise, $\exists\, a \in \{0,1\}^n$ such that

$$f(a) = 0 \text{ but } res_\beta(g_t)(a) = 1.$$

This means that each monomial in $DNF_\beta(g_t)$ contains at least one prime implicant of f.

Moreover, for each prime implicant p_i, $1 \leq i \leq r$ there is $j \in \{1, 2, \ldots, t_0\}$ with $m_j = p_i$. Otherwise, $\exists\, a \in \{0,1\}^n$ such that

$$f(a) = 1 \text{ but } res_\beta(g_t)(a) = 0.$$

Hence, we can restrict our considerations to the prime implicants contained in $DNF_\beta(g_t)$.

Each DNF-formula can be transformed into an equivalent CNF-formula. To see this let

$$\alpha := \bigvee_{i=1}^{t} m_i$$

be a DNF-formula which computes $f \in B_n$. To obtain an equivalent CNF-formula γ, we pick from each monomial m_i, $1 \leq i \leq t$ one literal and perform the disjunction of all chosen literals. Then, the conjunction of all clauses which can be constructed in this way is a CNF-formula

$$\gamma = \bigwedge_{j=1}^{s} d_j$$

which corresponds to the DNF-formula α. We call this transformation a DNF/CNF-switch.

Analogously, we can transform a CNF-formula into an equivalent DNF-formula (CNF/DNF-Switch).

Exercise:

a) Describe a CNF/DNF-Switch.

b) Let α be a DNF-formula (CNF-formula). Prove that the formula β obtained by a DNF/CNF-switch (CNF/DNF-switch) computes the same function as α.

continue at p. 117 →

Let $g_1, g_2, \ldots, g_t$ be the nodes of β numbered in any topological order. The function f is computed at the output node g_t. Starting with g_1, the DNF-representations $DNF_\beta(g_i)$, $1 \leq i \leq t$ are treated in this order.

Idea:
Replace $DNF_\beta(g_i)$ by an approximation $DNF'_\beta(g_i)$ such that we have the needed structural information.

After the replacement, $DNF_\beta(g_j)$, $j > i$ has to be updated such that for its construction $DNF'_\beta(g_i)$ is used instead of $DNF_\beta(g_i)$. Therefore, not the function f but an approximation f' of f is computed at the output node g_t. Hence there are inputs $c \in \{0,1\}^*$ such that $f'(c) \neq f(c)$. In the subsequence

$DNF_\beta(g_j)$ denotes
- the current function computed at the node g_j if $j > i$
- $DNF'_\beta(g_j)$ immediately before its replacement if $j \leq i$

where g_i is the last node for which $DNF_\beta(g_i)$ has been replaced by $DNF'_\beta(g_i)$.

Let f_1 (f_2) denote the function computed at the output node g_t after the approximation of $DNF_\beta(g_{i-1})$ ($DNF_\beta(g_i)$) and before the approximation of $DNF_\beta(g_i)$ ($DNF_\beta(g_{i+1})$). We say that the approximator of the node g_i introduces an error with respect to the

input $c \in \{0,1\}^n$ if $f_1(c) = f(c)$ but $f_2(c) \neq f(c)$.

Note that $f'(c) \neq f(c)$ implies that there exists a node g_ℓ in β such that the approximator of g_ℓ introduces an error with respect to the input c.

The approximators should be designed in a way such that the following is fulfilled:

1) After the replacement of $DNF_\beta(g_t)$, the number of inputs $c \in \{0,1\}^n$ with $f'(c) \neq f(c)$ is "large".

2) For all nodes g_i, $1 \leq i \leq t$, the number of inputs $c \in \{0,1\}^n$ where an error with respect to c is introduced by the replacement of $DNF_\beta(g_i)$ by $DNF'_\beta(g_i)$ is "small".

Note that these properties imply that a monotone network computing the function f has to be "large".

How to approximate DNF-formulas such that these properties are fulfilled?

The general idea is to bound the size of the monomials in the DNF-formulas constructed at the nodes of β. The size of a monomial can be its number of distinct literals or another measure.

Goal:
Development of a general framework for the definition of the size of a monomial.

The type $T(m)$ of a monomial is a set of certain elements used for the definition of the size of m. The type of a monomial has to fulfill the union property; i.e., for $m = m_0 m_1$ there holds

$$T(m) = T(m_0) \dot{\cup} T(m_1)$$

where $T(m_0) \dot{\cup} T(m_1)$ means the following

$$\begin{cases} a \in T(m_0) \Rightarrow a \in T(m) \text{ or } \exists b \in T(m_1): a \cup b \in T(m) \\ a \in T(m_1) \Rightarrow a \in T(m) \text{ or } \exists b \in T(m_0): a \cup b \in T(m). \end{cases}$$

If $T(m) = T(m_0) \cup T(m_1)$ then the type of a monomial fulfills the strong union property.

Then, the size of the monomial m is a function of the cardinality of its type $T(m)$. Often, the size of a monomial is the cardinality of its type.

Example:

a) If the size of a monomial is its length then we use $T(m)$ to be the set of variables contained in m. Such a type is called v-type. Then, the size of a monomial m is the cardinality of its type $T(m)$.

b) With respect to the clique function, a more complicated measure is usual. We say that a monomial m touches a node $v \in V$ iff there is at least one variable in m corresponding to an edge which has end node v. Then, the size of

the monomial m is the number of different nodes in V which are touched by m. To get this, we use T(m) to be the set of nodes in V touched by m. We call this type n-type. Again, the size of a monomial m is the cardinality of its type T(m).

Note that v-types and also n-types fulfill the strong union property.

Let r be the upper bound for the size of a monomial in an approximator with respect to a node g_i. An obvious way to bound the size of the monomials would be the following:

- For the construction of the approximator $DNF'_\beta(g_i)$ construct the DNF-representation $DNF_\beta(g_i)$ of the current function computed at g_i and remove each monomial of size larger than r.

To explain the approximation method, we assume that the size of a monomial m is the cardinality of its type T(m) which fulfills the strong union property.

For input nodes, the approximator and the original DNF-representation are the same. For the comparision of the approximator $DNF'_\beta(g_i)$ and the DNF-representation $DNF_\beta(g_i)$ of the current function computed at g_i immediately before its approximation suppose that g_{i_1} and g_{i_2} are the direct predecessors of the gate g_i.

By construction, each monomial in

$$DNF'_\beta(g_{i_1}) = \bigvee_{j=1}^{t_1} m_j \quad \text{and} \quad DNF'_\beta(g_{i_2}) = \bigvee_{e=1}^{t_2} m'_e$$

has size at most r. We distinguish two cases.

Case 1: g_i is an $\vee$-gate.

Then each monomial in $DNF_\beta(g_i)$ has size at most r. Hence, we define

$$DNF'_\beta(g_i) := DNF_\beta(g_i).$$

$\Rightarrow$

No error is introduced by the approximator

(1)

Case 2: g_i is an $\wedge$-gate.

Then

$$DNF_\beta(g_i) = \bigvee_{j=1}^{t_1} \bigvee_{e=1}^{t_2} (m_j \wedge m'_e).$$

To obtain $DNF'_\beta(g_i)$, we remove from $DNF_\beta(g_i)$ all monomials $m_j m'_e$ of size larger than r.

Strong union property $\Rightarrow$

At least one of m_j and m'_e has size larger than $\frac{r}{2}$.

$\Rightarrow$

An input $c \in \{0,1\}^n$ for which the removal of $m_j m'_e$ introduces an error must fulfill the monomial $m_j m'_e$ and hence, a monomial of size larger than $\frac{r}{2}$.

This can be used to get an upper bound for the number of inputs for which an error could be introduced because of the removal of the monomial $m_i; m_i'$.

If the number of monomials which are removed would be small enough then perhaps, we could prove an upper bound for the number of errors which could be introduced by the approximation at an $\wedge$-gate which is small enough.

How to get such an upper bound? First of all, we can restrict us to the minimal monomials in $DNF_B(g_i)$ which are not submonomial of another monomial in $DNF_B(g_i)$ since this does not change the function computed by $DNF_B(g_i)$.

But at an $\vee$-gate, the number of minimal monomials in a DNF-formula could be increased.

↝

We need a mechanism which bounds the number of minimal monomials which could be removed during the construction of an approximator. Two such mechanisms are known:

1) Razborov-approximators, introduced by Alexander Razborov in 1984 and
2) CNF-DNF-approximators, introduced by implicitly by Haken and explicitly by Jukna, by Berg and Ulfberg and by Amano and

Maruoka ten years later.

Since CNF-DNF-approximators are simpler, we shall describe these first.

5. CNF-DNF-approximators

References:

- Armin Haken, Counting bottlenecks to show monotone P≠NP, 36th FOCS (1995), 36-40.
- Stasys Jukna, Combinatorics of monotone computations, Combinatorica 19 (1999), 65-85.
- Christer Berg, Staffan Ulfberg, Symmetric approximation arguments for monotone lower bounds without sunflowers, Comput. Complexity 8 (1999), 1-20.
- Kazuyuki Amano, Akira Maruoka, The potential of the approximation method, SIAM J. Comput. 33 (2004), 433-447.
- Stasys Jukna, Boolean Function Complexity: Advances and Frontiers, Springer 2012, 257-263.

In 1995, Armin Haken has introduced the so-called "bottleneck counting method" to prove an exponential lower bound for the monotone network complexity of a Boolean function which resembles the clique function. Instead of approximating the behaviour of the network directly, he has defined a function μ which

maps inputs to the gates of the network. He has shown that the total number of inputs mapped to the gates by μ has to be "large" and also that only "few" inputs are being mapped to each gate. Hence, the network must contain "many" gates. Shortly after that Jukna, Berg and Ulfberg and also Amano and Maruoka have observed that Haken's approach is indeed an approximation argument in disguise. The have translated Haken's proof into an approximation proof which uses CNF-DNF-approximators.

5.1 The method

We will describe CNF-DNF-approximators. Before doing this, we consider the general features of the approximation method again.

Let $f \in B_n$ be the monotone Boolean function for which we intend to prove a large lower bound of its monotone complexity. Let β be a monotone network computing f and let $g_1, g_2, \ldots, g_L$ be the nodes of β numbered in any topological order. During the approximation process, let g_i be the considered node.

To obtain $DNF'_\beta(g_i)$, the approximation method removes from $DNF_\beta(g_i)$ each monomial of size larger than a certain bound r.

$\Rightarrow$

Only for inputs $c \in f^{-1}(1)$, an error could be introduced.

Therefore, to get a large lower bound for the monotone complexity of f, the function f must have the following property:

P1: Only "few" inputs in $f^{-1}(1)$ fulfill a monomial of size larger than r.

The approximation method has to take care that only "few" monomials are removed to obtain $DNF'_B(g_i)$ from $DNF_B(g_i)$.

To get this feature, CNF-DNF-approximators use the following implementation of a CNF/DNF-switch:

Let $d_1, d_2, \ldots, d_s$ be the clauses of the CNF-formula for which we intend to construct an equivalent DNF-formula. The CNF/DNF-switch can be organized as the construction of a tree T in the following way:

(1) Each edge in T is labelled by a variable. With each node w in T we associate the monomial $m(w)$ which is obtained by the conjunction of the variables on the unique path from the root of T to w. The tree T is constructed while expanding $d_0, d_1, \ldots, d_s$ where d_0 is the empty clause.

(2) While expanding d_0, the root of T is created. The associated monomial is the empty monomial.

(3) Suppose that w is a leaf that was created while expanding d_i. Then the clause d_{i+1} is expanded

at the leaf w in the following way: If m(w) contains already a variable which is in the clause d_{i+1} then each input which fulfills m(w) already fulfills the clause d_{i+1}. Hence, no expansion with respect to d_{i+1} is needed such that the next clause d_{i+2} has to be expanded at the leaf w. Otherwise, the leaf w obtains for each variable in d_{i+1} a new son w'. The edge (w, w') is labelled with the corresponding variable.

An important property of the construction above is that the number of variables in m(w) is equal the length of the path from the root of T to the node w.

If the size of a monomial is its length then we would obtain a DNF-formula with each monomial is of size at most r by removing all paths in T which have length larger than r. Each monomial corresponding to a removed path has a monomial corresponding to a path of length r which starts in the root of T as a submonomial.

$\Rightarrow$

Each input $c \in f^{-1}(1)$ for which the removal of the paths has introduced an ~~error~~ must fulfill a monomial corresponding to a path of length r which starts in the root of T.

$\Rightarrow$

A small upper bound for the number of such monomials would be desirable.

Observation:
If the length of each clause d_i, $1 \leq i \leq s$ is at most k then we would obtain the obvious upper bound k^r for the number of such monomials.

$\rightsquigarrow$

Idea
Approximate for each node g_i, $1 \leq i \leq t$

- $DNF_\beta(g_i)$ by a DNF-formula $DNF'_\beta(g_i)$ which contains only monomials of size $\leq r$.
- $CNF_\beta(g_i)$ by a CNF-formula $CNF'_\beta(g_i)$ which contains only clauses of size $\leq k$.

The approximation of the CNF-formulas is dual to the approximation of the DNF-formulas.

To obtain $CNF'_\beta(g_i)$, the approximation method removes from $CNF'_\beta(g_i)$ each clause of size larger than the bound k.

$\Rightarrow$
Only for inputs $c \in f^{-1}(0)$, an error could be introduced.

Therefore, to get a large lower bound for the monotone complexity of f, the function f must have the following property:

P2: Only "few" inputs in $f^{-1}(0)$ falsify a clause of size larger than k.

In the same way as the approximation of DNF-formulas, we obtain the upper bound r^k for the number of such clauses.

To get a large lower bound for the monotone complexity of f, the number of errors which appear because the approximation of the DNF- or the CNF-representation of $res_\beta(g_t)$ has to be large. Therefore, the function f must have the following additional property:

P3:

- If $CNF'_\beta(g_t)$ is not the constant function "one" then "many" inputs in $f^{-1}(1)$ do not fulfill $CNF'_\beta(g_t)$

or

- If $DNF'_\beta(g_t)$ is not the constant function "zero" then "many" inputs in $f^{-1}(0)$ do not falsify $DNF'_\beta(g_t)$.

Because of properties P1 and P2, the third property implies a large lower bound.

Instead of using the whole sets $f^{-1}(1)$ and $f^{-1}(0)$, one can use more appropriate subsets $T_1 \subseteq f^{-1}(1)$ and $T_0 \subseteq f^{-1}(0)$ to prove the lower bound.

We call a CNF/DNF (DNF/CNF) - switch where all monomials (clauses) of size larger than $r(k)$ are removed a <u>CNF/DNF (DNF/CNF) - approximator switch</u>.

Now we are prepared to give a precise description of the CNF-DNF - approximation process. The nodes of a network β which computes the function f are considered in any topological order $g_1, g_2, \ldots, g_t$. For each node g_i, the formulas $CNF_\beta(g_i)$ and $DNF_\beta(g_i)$ are approximated by the formulas $CNF'_\beta(g_i)$ and $DNF'_\beta(g_i)$. We distinguish three cases.

<u>Case 1</u>: g_i is an input node.

Then
$$CNF'_\beta(g_i) := CNF_\beta(g_i)$$
and
$$DNF'_\beta(g_i) := DNF_\beta(g_i).$$

<u>Case 2</u>: g_i is an $\wedge$-gate.

Let g_{i_1} and g_{i_2} be the direct predecessors of g_i. Then
$$CNF'_\beta(g_i) := CNF'_\beta(g_{i_1}) \wedge CNF'_\beta(g_{i_2}).$$

Since the size of each clause in $CNF'_\beta(g_{i_1})$ and in $CNF'_\beta(g_{i_2})$ is at most k, each clause in $CNF'_\beta(g_i)$ has also at most the size k.

$DNF'_\beta(g_i)$ is obtained from $CNF'_\beta(g_i)$ by

- performing a CNF/DNF-approximator switch.

Case 3: g_i is an $\vee$-gate.

Let g_{i_1} and g_{i_2} be the direct predecessors of g_i. Then

$$DNF'_\beta(g_i) := DNF'_\beta(g_{i_1}) \vee DNF'_\beta(g_{i_2}).$$

Since the size of each monomial in $DNF'_\beta(g_{i_1})$ and in $DNF'_\beta(g_{i_2})$ is at most r, each monomial in $DNF'_\beta(g_i)$ has also at most the size r.

$CNF'_\beta(g_i)$ is obtained from $DNF'_\beta(g_i)$ by

- performing a DNF/CNF-approximator switch.

Now we are prepared to apply CNF-DNF-approximators to concrete Boolean functions.

5.2 Applications

Let $CLIQUE(m,s)$ be the Boolean function of $n := \binom{m}{2}$ variables representing the edges of an undirected graph $G = (V,E)$ on m nodes whose value is one iff G contains a clique of size s. In the subsequence, we use the node set $V = \{1,2,\dots,m\}$. Furthermore, x_{ij} denotes

the variable which corresponds to the edge (i,j). We will identify the edge (i,j) and the variable x_{ij}. In the subsequence, f denotes the function CLIQUE(n,s).

First, we will specify the size of a monomial and the size of a clause. The <u>size of a monomial m</u> is the number of different nodes touched by m. For the definition of the size of a clause d, we consider the graph $G(d) := (V, E(d))$ where $E(d)$ contains exactly those edges which correspond to the variables in d. The <u>size of the clause d</u> is m minus the number of connected components in $G(d)$.

For the approximators $DNF'_\beta(g)$ and $CNF'_\beta(g)$, we use the values

$$r := \lfloor \sqrt{s} \rfloor \quad \text{and} \quad k := \left\lfloor \frac{n}{8s} \right\rfloor.$$

Since $r = \lfloor \sqrt{s} \rfloor$, less than $\lfloor \sqrt{s} \rfloor$ different nodes in V can be touched by a monomial in $DNF'_\beta(g)$. Therefore, the number of variables in such a monomial is bounded by $\frac{r^2}{2} \leq \frac{s}{2}$.

$k = \lfloor \frac{n}{8s} \rfloor$ implies that a graph corresponding to a clause in $CNF'_\beta(g)$ has more than $n - \lfloor \frac{n}{8s} \rfloor$ connected components. Since each connected component contains at least one node, at most

$$2k \leq \frac{n}{4s}$$

nodes are touched by a clause in $CNF'_\beta(g)$.

Next, we define appropriate subsets $T_1 \subseteq f^{-1}(1)$ and $T_0 \subseteq f^{-1}(0)$. For an implicant m of f, the input $I_1(m)$ is defined by

$$I_1(m) = (a_1, a_2, \ldots, a_n) \text{ where } a_i = 1 \text{ iff } m \leq x_i.$$

This means that $I_1(m)$ assigns the value one to a variable x_i iff x_i is contained in m.

For an f-clause d of f, the input $I_0(d)$ is defined by

$$I_0(d) = (b_1, b_2, \ldots, b_n) \text{ where } b_i = 0 \text{ iff } x_i \text{ is contained in } d.$$

Then we set

$$T_1 := \{ I_1(p) \mid p \in PIM(f) \}.$$

For the definition of T_0, we consider colourings $h: V \to \{1, 2, \ldots, s-1\}$ of the nodes in V by $s-1$ colours. The graph

$$G(h) := (V, E(h))$$

corresponding to the colouring h contains all edges between two nodes in different colour classes and no edge between two nodes in the same colour class.

$\Rightarrow$

$G(h)$ is a complete ℓ-partite graph where $\ell \in \{1, 2, \ldots, s-1\}$ is the number of colours used by h.

The clause $d(h)$ corresponding to the colouring h contains exactly those variables x_{uv} with both nodes u and v are coloured with the same colour; i.e., $h(u) = h(v)$.

$\ell \leq s-1 \Rightarrow$

Each s-clique must contain at least two nodes which are in the same colour class.

$\Rightarrow$

$d(h)$ is an f-clause.

Now we are prepared to define T_0.

$T_0 := \{ I_0(d(h)) \mid h: V \to \{1,2,\ldots,s-1\}$ is a colouring$\}$.

Next we will adapt the CNF/DNF- and DNF/CNF-approximator switches such that the length of the paths from the root to the nodes in T where only edges descending from a node with more than one son are counted is equal to the size of the corresponding monomial and clause, respectively.

CNF/DNF-approximator switch

Let $d_1, d_2, \ldots, d_\ell$ be the clauses in $CNF'_\beta(g_i)$ given in any fixed order and let d_0 be the empty clause. Assume that w is a leaf which was created while expanding d_j.

Treatment of d_{j+1} w.r.t. w

We call a variable x_{uv} tight for a monomial m iff both end nodes u and v are touched by m. We distinguish two cases:

Case 1: d_{j+1} contains a variable x_{uv} which is tight for $m(w)$.

For each $a \in T_1$ which satisfies $m(w)$, both nodes u and v have to be contained in the clique which corresponds to $p \in PiM(f)$ with $a = I_1(p)$.

$\Rightarrow \quad x_{uv}(a) = 1.$

We create only one son w' for w and label the edge (w, w') with x_{uv}.

Case 2: d_{j+1} contains no variable which is tight for $m(w)$.

Then the variables in d_{j+1} separates into the following two sets:

$$Var_0 := \{ x_{uv} \mid \text{both end nodes } u \text{ and } v \text{ are not touched by } m(w) \}.$$

$$Var_1 := \{ x_{uv} \mid \text{exactly one of } u \text{ and } v \text{ is touched by } m(w) \}.$$

First we will consider the variables in Var_1. Let

$$V' := \{ u \in V \mid u \text{ is not touched by } m(w) \text{ but } \exists\, x_{uv} \in Var_1 \}$$

For each $u \in V'$ let

$$N(u) := \{ v \in V \mid x_{uv} \in Var_1 \}.$$

For each $u \in V'$, we choose any $v \in N(u)$ and create a son w_u of the node w. The edge (w, w_u) is labelled with the variable x_{uv} and we define that the edge (w, w_u) is touched by the node u.

This suffices since two monomials touching the same nodes are submonomials of the same prime implicants of the clique function. Since d_{j+1} touches at most $2k$ nodes, at most

$$2k \leq \frac{m}{4s}$$

sons are created.

Now we will consider the variables in Var_0. As long as there is an edge corresponding to a variable in Var_0 such that none of its two end nodes is chosen, we choose an end node u of such an edge and create a son w'_u for w. The corresponding edge (w, w'_u) obtains no label. We define that the edge (w, w'_u) is touched by the node u. Since d_{j+1} touches at most $2k$ nodes, at most

$$2k \leq \frac{m}{4s}$$

sons are created. Let

$$V'' := \{ u \in V \mid w'_u \text{ is created} \}$$

and for each $u \in V''$

$$N'(u) := \{ v \in V \mid x_{uv} \in Var_0 \}.$$

For each $u \in V''$ for each $v \in N'(u)$, we create a son w''_v of the node w'_u and label the edge (w'_u, w''_v) with the variable x_{uv}. We define that the node v touches the edge (w'_u, w''_v). Again, since d_{j+1} touches at most $2k$ nodes, at most

$$2k \leq \frac{m}{4s}$$

sons for w'_u are created.

After the construction of T, for each edge descending from a node with more than one son, there is a node in V which touches the edge. By construction, no two edges on a path from the root of T to a leaf are touched by the same node.

$\Rightarrow$

For each leaf w, the size of $u(w)$ is equal the number of nodes touching the edges on the path from the the root of T to the leaf w.

By removing all monomials of size larger than r, we obtain $DNF'_\beta(g_i)$.

Exercise:

Show that the number of inputs in T_1 for which $DNF'_\beta(g_i)$ could introduce an error is bounded by $\binom{m-r}{s-r}\left(\frac{m}{4s}\right)^r$.

DNF/CNF - approximator switch

Let $m_1, m_2, \ldots, m_\ell$ be the monomials in $DNF'_\beta(\varphi_i)$ given in any fixed order and let m_0 be the empty monomial. Suppose that w is a leaf which was created while expanding m_j.

Treatment of m_{j+1} w.r.t. w:

A variable x_{uv} in m_{j+1} is called good iff both end nodes of the edge (u,v) are contained in the same connected component of the graph

$$G(d(w)) = (V, E(d(w))).$$

By construction, each input $b \in T_0$ which falsifies $d(w)$ has the property that each connected component of $G(d(w))$ is contained in one colour class with respect to the colouring h where $b = I_0(d(h))$.

$\Rightarrow$ b must falsify each good variable as well.

We distinguish two cases.

Case 1: m_{j+1} contains a good variable x_{uv}.

Since each input $b \in T_0$ which falsifies $d(w)$ also falsifies the variable x_{uv}, $b_{uv} = 0$.

$\Rightarrow$

It suffices to create one son w' of w and to label the edge (w,w') with x_{uv}.

Case 2: m_{j+1} contains no good variable.

Then each variable x_{uv} in m_{j+1} connects two connected components of $G(d(w_j))$. The leaf w obtains for each variable x_{uv} in m_{j+1} a new son w'. The edge (w, w') is labelled with x_{uv}.

By construction, when descending on an edge to a son from a node with more than one son, the number of connected components of the associated graph decreases by one.

$\Rightarrow$

The size of the corresponding clause increases by one.

$\Rightarrow$

There are at most k such nodes on a path from the root to a node w with $d(w_j)$ has exactly the size k.

Since each monomial in $DNF'_B(g_i)$ contains at most $\frac{s}{2}$ variables, each node in T has at most degree $\frac{s}{2}$. Hence,

$$\leq \left(\frac{s}{2}\right)^k$$

nodes in T with the corresponding clause has exactly size k exist.

Note that different colourings can yield the same input in T_0. For counting properties, it would be simpler to consider such inputs as different. Then $|T_0| = (s-1)^m$.

Exercise:
Show that the number of inputs in T_0 for which $CNF'_\beta(g_i)$ could introduce an error is bounded by $(\frac{s}{2})^{\frac{1}{2}} \cdot (s-1)^{m-\frac{1}{2}}$.

Exercise:
Show that either $CNF'_\beta(g_t)$ computes the constant function one or $CNF'_\beta(g_t)$ computes the value of at least half of the inputs in T_1 incorrectly.

Altogether, we obtain the following theorem.

Theorem 5.1

Let $s \leq m^{2/3}$. Then $C_{\Omega_m}(CLIQUE(m,s)) \geq 2^{\Omega(\sqrt{s})}$.

Proof:

Exercise

■

Next, we will apply CNF-DNF-approximators to a function for which Alexander Andreev could prove the first exponential lower bound for the monotone complexity of a Boolean function in NP.

Andreev's function is the characteristic function $POLY(q,s)$ of the following problem:

For a prime power $q \geq 2$ let $GF(q)$ denote the finite field with q elements. Let $G = (A, B, E)$ be a bipartite graph where $A := GF(q)$ and $B := GF(q)$.

For given q and s, the problem is to decide whether there exists a polynomial p over $GF(q)$ of degree at most $s-1$ such that for all $i \in A$ there holed $(i, p(i)) \in E$.

$POLY(q,s)$ is a monotone Boolean function of $n := q^2$ variables. Given any polynomial p over $GF(q)$ of degree $\leq s-1$, it is easy to check if for all $i \in A$, $(i, p(i)) \in E$. Hence, $POLY(q,s) \in NP$.

The prime implicants of $POLY(q,s)$ are the monomials corresponding exactly to a polynomial p over $GF(q)$ of degree $\leq s-1$; i.e., $\bigwedge \{x_{i,p(i)} \mid 0 \leq i \leq q-1\}$

In the subsequence, f denotes the function $POLY(q,s)$. Again we use

$$T_1 := \{ I_1(p) \mid p \in PIM(f) \}.$$

Instead of using a subset T_0 of $f^{-1}(0)$, an input $a \in \{0,1\}^n$ is randomly constructed where $a_{ij} = 1$, $i,j \in GF(q)$ with probability $1-\varepsilon$ for $\varepsilon = \frac{2s \ln q}{q}$.

The following lemma shows that the probability for $a \in f^{-1}(1)$ is small.

Lemma 5.1

Let $a \in \{0,1\}^n$ be randomly constructed by choosing $a_{ij} = 1$ with probability $1-\varepsilon$ for $\varepsilon = \frac{2s \ln q}{q}$ for all $i,j \in GF(q)$. Then the probability that $a \in f^{-1}(1)$ is at most q^{-s}!

Proof:

The probability that q specific edges corresponding to a given polynomial are present in a randomly chosen input is

$$(1-\varepsilon)^q.$$

$\Rightarrow$

The probability that the edges corresponding to at least one of the q^s possible polynomials are present in a randomly chosen input is

$$\leq q^s(1-\varepsilon)^q \leq q^s e^{-\varepsilon q} = q^s q^{-2s} = q^{-s}.$$

■

The size of a monomial or a clause is its length. We use the following upper bounds r and k for the sizes of a monomial and a clause.

$$r := s \quad \text{and} \quad k := \frac{1}{2} \cdot q^{2/3}.$$

Here, no adaption of the CNF/DNF- and DNF/CNF-approximator switches is necessary. Again, β is a monotone Boolean network computing f and $g_1, g_2, \ldots, g_t$ are the nodes of β in any topological order.

Lemma 5.2

$CNF'_\beta(g_t)$ computes the constant function one or $CNF'_\beta(g_t)$ computes the value of at least half of the inputs in T_1 incorrectly.

Proof:

Assume that $CNF'_\beta(g_t)$ does not compute the constant function one. Then there exists an input $b \in \{0,1\}^u$ such that $CNF'_\beta(g_t)(b) = 0$.

$\Rightarrow$

$CNF'_\beta(g_t)$ contains a clause d.

Each $a \in T_1$ with $CNF'_\beta(g_t)(a) = 1$ has to fulfill the clause d; i.e., the corresponding prime implicant contains a variable in d.

Containing such a variable is equivalent to fixing the value of a polynomial at one point.

There are q^{s-1} such polynomials and therefore q^{s-1} such prime implicants. Furthermore, d contains at most $k = \frac{1}{2} q^{2/3}$ variables.

$\Rightarrow$

$$\leq k \cdot q^{s-1} = \frac{1}{2} q^{2/3} \cdot q^{s-1} < \frac{1}{2} q^s.$$

Since $|T_1| = q^s$, the assertion follows. ∎

The next two lemmas show that only few errors are introduced at a gate g_i because of the approximation performed with respect to g_i.

Lemma 5.3

At an $\wedge$-gate g_i, the replacement of $DNF_\beta(g_i)$ by $DNF'_\beta(g_i)$ introduces an error for at most k^r inputs in $\overline{T_\wedge}$.

Proof:

Since $CNF'_\beta(g_i) = CNF_\beta(g_i)$, the replacement of $CNF_\beta(g_i)$ by $CNF'_\beta(g_i)$ does not introduce any error.

As observed above, a CNF/DNF-approximator switch constructs at most k^r monomials of size r.

Each input $\alpha \in T_1$ for which an error is introduced by the replacement of $DNF_\beta(g_i)$ by $DNF'_\beta(g_i)$ must fulfill such a monomial.

Since a polynomial of degree $\leq s-1$ is completely specified by its value at s different points and $r = s$, a monomial of size r is fulfilled by at most one input.

$\Rightarrow$

For at most k^r inputs in $\overline{T_1}$, an error is introduced by the replacement of $DNF_\beta(g_i)$ by $DNF'_\beta(g_i)$.

■

Lemma 5.4

At an $\vee$-gate g_i, the replacement of $CNF_\beta(g_i)$ by $CNF'_\beta(g_i)$ introduces an error for a random input $a \in \{0,1\}^n$ with probability at most $(r \cdot \varepsilon)^k$.

Proof:

Since $DNF'_\beta(g_i) = DNF_\beta(g_i)$, the replacement of $DNF_\beta(g_i)$ by $DNF'_\beta(g_i)$ does not introduce an error.

A DNF/CNF - approximator switch applied to $DNF'_\beta(g_i)$ constructs at most r^k clauses of size k.

Each input a for which an error is introduced by the removal of such a clause must falsify such a clause. The probability that a random input falsify any given variable is ε.

$\Rightarrow$

The probability that k given variables are falsified by a random input is ε^k

$\Rightarrow$

The probability that an error is introduced for a random input $a \in \{0,1\}^n$ is

$$\leq (r \cdot \varepsilon)^k.$$

■

Now we are prepared to prove the following theorem.

Theorem 5.2

Let $s \leq \frac{1}{2}\sqrt{\frac{q}{\ln q}}$. Then $C_{\Sigma_m}(\mathrm{POLY}(q,s)) \geq q^{\Omega(s)}$

Proof:

Exercise ∎

Resume:

Applying CNF-DNF-approximators to a monotone function f, the function f must have the following three properties to prove a large lower bound:

P1: Only "few" inputs in $f^{-1}(1)$ fulfill a monomial of size larger than r.

P2: Only "few" inputs in $f^{-1}(0)$ falsify a clause of size larger than k.

P3: A clause of size $\leq k$ is fulfilled by only a fraction d_1 of the inputs in $f^{-1}(1)$, $0 < d_1 < 1$

or

A monomial of size $\leq r$ is falsified by only a fraction d_0 of the inputs in $f^{-1}(0)$, $0 < d_0 < 1$.