

CONCRETE COMPLEXITY THEORY

WINTER TERM '08, LECTURE GIVEN BY PROF. DR. F. MEYER AUF DER HEIDE

Preface

These lecture notes cover (hopefully the most of) the course CONCRETE COMPLEXITY THEORY, lectured by Prof. Dr. F. Meyer auf der Heide at the University of Paderborn, Germany. They are neither complete nor fully proofread, let alone being far from uniformly well-written.

If you stumble upon any kind of mistake, feel free to send me an eMail (carsten@caero.de). The \LaTeX -sources of this notes can be accessed through http://blog.caero.de/?page_id=198 and modified according to your needs - under one condition: Republish the sources, 'caue "human knowledge belongs to the world!" :-)

Carsten Rösnick, 8th April 2009

Contents

1	Boolean Circuits	1
1.1	Circuit Complexity	1
1.2	Formula Complexity	3
1.3	Monotone Circuits	8
1.4	Unbounded Fan-In Circuits	9
1.5	A lower bound for Parity _n	11
2	Computations over the Reals and over the Integers	13

Bibliography

[M. Ben-Or] *Lower Bounds For Algebraic Computation Trees*, 1983

[F. Meyer auf der Heide] *Concrete Complexity Theory, Lecture notes and slides*,
<http://www.hni.uni-paderborn.de/alg/lehre/ws-20082009/concrete-complexity-theory>, PB 2008

[I. Wegener] *The Complexity of Boolean Functions*, Wiley, Dortmund 1991

[U. Zwick] *Concrete Complexity*, <http://www.cs.tau.ac.il/~zwick/CS277.html>, Tel Aviv 1997

Chapter 1

Boolean Circuits

1.1 Circuit Complexity

Definition 1.1.1. Define $B_n^m := \{f : \{0, 1\}^n \rightarrow \{0, 1\}^m\}$ as the set comprised of all boolean functions with n inputs and m outputs. Then $B_1 := B_1^1$ consists of the operations $x \mapsto 0, 1, x, \bar{x}$.

Remark. $|B_n^m| = |\text{image}|^{|\text{preimage}|} = 2^{m2^n}$

Definition: $x^1 := x, x^0 := \bar{x}$

Definition 1.1.2 (Formulae). We call a circuit with fan-out ≤ 1 a formula, i.e. formulae are binary trees.

Definition 1.1.3. Denote by $\text{size}(\mathcal{C})$ the number of gates of the circuit \mathcal{C} .

Circuit complexity $C_\Omega(f) := \min\{\text{size}(\mathcal{C}) \mid \mathcal{C} \text{ is a circuit over basis } \Omega \text{ for } f\}$
Formula complexity $L_\Omega(f) := \min\{\text{size}(\mathcal{F}) \mid \text{Formula } \mathcal{F} \text{ over } \Omega \text{ computes } f\}$
Depth complexity $D_\Omega(f) := \text{minimum depth required to compute } f \text{ over } \Omega$

Definition 1.1.4 (Min- and maxterms). A minterm m_a for $a = (a_1, \dots, a_n) \in \{0, 1\}^n$ is defined as $m_a(x) = x_1^{a_1} \wedge \dots \wedge x_n^{a_n}$. Hence, On the other hand, a maxterm is defined as $s_a(x) = \neg m_a(x) = \neg x_1^{a_1} \vee \dots \vee \neg x_n^{a_n} = x_1^{1-a_1} \vee \dots \vee x_n^{1-a_n}$.

Obviously, every function $f \in B_n$ can be expressed as

$$f(x) = \bigvee_{a \in f^{-1}(1)} m_a(x) = \bigwedge_{a \in f^{-1}(0)} s_a(x).$$

As we can see from the definition, minterms denote exactly the lines in the truth table of f for which $f(x) = 1$ holds. In case of $f(x) = 1$ there must be a minterm m_a such that $m_a(x) = 1$. Since we are choosing a as an element of f 's preimage we take all minterms into account. Hence, if $f(x) = 1$ is true then there is an $a \in f^{-1}(1)$ s.t. the corresponding minterm m_a is true, i.e. $m_a(x) = 1$.

Theorem 1.1.1 (Ring Sum Expansion). For each $f \in B_n$ there exists exactly one vector $(a_I)_{I \in 2^{\{1, \dots, n\}}}$ with $a_I \in \{0, 1\}$ such that

$$f(x_1, \dots, x_n) = \bigoplus_{I \in 2^{\{1, \dots, n\}}} a_I \wedge \bigwedge_{i \in I} x_i \tag{1.1}$$

holds.

Proof. Obviously $a^b = a \oplus b \oplus 1$ holds for each $a, b \in \{0, 1\}$. We define φ as a bijection of the form

$$\varphi : \{0, 1\}^n \xrightarrow{\sim} 2^{\{1, \dots, n\}} \text{ with } (c_1, \dots, c_n) \mapsto \{i \mid c_i = 1\}.$$

To prove the unique representation proposed in (1.1) we arbitrarily choose a boolean function $f \in B_n$ and write f as a polynomial of the form

$$f(x_1, \dots, x_n) = \bigoplus_{(c_1, \dots, c_n) \in \{0, 1\}^n} f(c_1, \dots, c_n) \wedge \bigwedge_{i=1}^n (x_i \oplus c_i \oplus 1).$$

By evaluating the polynomial at (x_1, \dots, x_n) we observe that all terms will be zero except the one with $x_i = c_i$ for all $i = 1, \dots, n$ and we get $f(c_1, \dots, c_n) = f(x_1, \dots, x_n)$ as the result. Hence we get

$$f(x_1, \dots, x_n) = \bigoplus_{(c_1, \dots, c_n) \in \{0, 1\}^n} a_{\varphi((c_1, \dots, c_n))} \wedge \bigwedge_{i \in \varphi((c_1, \dots, c_n))} x_i = \bigoplus_{I \in 2^{\{1, \dots, n\}}} a_I \wedge \bigwedge_{i \in I} x_i.$$

The a_I 's are uniquely defined for each $f \in B_n$. Since a boolean function is uniquely defined by its operation on the set of inputs and each input $x \in \{0, 1\}^n$ corresponds to exactly one tuple (c_1, \dots, c_n) (as we have seen above) the sequence of bits $(a_I)_{I \in 2^{\{1, \dots, n\}}}$ is uniquely defined for a chosen $f \in B_n$.

Remark: It is correct to exclude the x_i with $i \notin I$ from the \wedge -product because a_I is uniquely defined and the \wedge -product yields true as well. \square

Proposition 1.1.2. *RSE, DNF and CNF of Majority(x) have exponential size.*

Definition 1.1.5. $\text{Majority}(x) = 1 \iff \sum_{i=1}^n x_i \geq \frac{1}{2}, x = (x_1, \dots, x_n) \in \{0, 1\}^n$

Theorem 1.1.3. *Let $\Omega_1, \Omega_2 \subseteq B_1 \cup B_2$ be arbitrary complete bases. Then for each $f \in B_n$, $C_{\Omega_1}(f) = \mathcal{O}(C_{\Omega_2}(f))$ and $D_{\Omega_1}(f) = \mathcal{O}(D_{\Omega_2}(f))$ holds.*

Theorem 1.1.4 (Lupanov). $\forall f \in B_n : C_{U_2}(f) = (1 + o(1))2^n/n$.

Proof. (1) Observe $f(x) = (f_1(x) \wedge x) \vee (f_0(x) \wedge \bar{x})$

(2) Set $C(k = \# \text{free variables}) = \# \text{gates to calculate } f_k$ and conclude $C(n) \leq 2 \cdot C(n-1) + 3, C(2) = 1$. Thus $C(n) \leq 2^k \cdot C(n-k) + 3(2^k - 1)$.

(3) Going $n-k$ levels up from the root results in at most 2^{n-k} subfunctions $f_i \in B_k$. Furthermore, there are at most $3(2^{n-k} - 1)$ gates up to level $n-k$ ($2^{n-k} - 1$ nodes and 2 AND-gates plus 1 OR-gate per node).

(4) Observe: There exists some k such that $2^{2^k} < 2^{n-k}$, i.e. it's cheaper to generate all functions $f_i \in B_k$ than to further enroll the circuit by above's method.

(5) $2 \cdot 2^{2^{k-1}}$ AND-gates and $(2^{2^{k-1}})^2$ OR-gates on level $n-k$

$$\begin{aligned} |B_{k-1}| = 2^{2^{k-1}} &\implies (2^{2^{k-1}})^2 = 2^{2^k} \text{ combinations of functions } f_i \in B_{k-1} \\ &\implies 2^{2^k} \text{ OR-gates at level } n-k. \end{aligned}$$

(6) Summing up leads us to the proposed bound. Set $k := \lfloor \log(n - \log n) \rfloor$. Then:

$$\begin{aligned} A(k) &:= \# \text{gates to calculate all functions } f_i \in B_{k-1} = A(k-1) + 2 \cdot 2^{2^{k-1}} + 2^{2^k} = (1 + o(1))2^{2^k} \\ \implies \# \text{gates to compute } f \in B_n &\leq \underbrace{(1 + o(1))2^{2^k}}_{=A(k)} + \underbrace{3(2^{n-k})}_{\geq C(n-k)} = (1 + o(1))\frac{2^n}{n} + 3\frac{2^n}{n - \log n} = (4 + o(1))\frac{2^n}{n}. \end{aligned}$$

\square

Theorem 1.1.5 (Shannon). *There are functions $f \in B_n$ with $C_{B_2}(f) \geq 2^n/n$.*

Proof. (1) Count the number of circuits with $\leq s$ gates and n inputs, denoted by $C(n, s)$.

$$C(n, s) \leq \underbrace{(|B_2| \cdot \underbrace{(n+s)^2}_{\substack{\text{two inputs, hence} \\ (n+s)^2 \text{ sources to} \\ \text{choose } (x, y) \text{ from}}})^s}_{s \text{ gates to specify the input} \\ \text{tuple as well as the operation for}} \cdot \underbrace{\frac{s}{s!}}_{(*)}$$

The factor in $(*)$ has to be there since we are not interested in permutations of the gates – except the output gate. Hence, by fixing the output gate and throwing away the permutations of the remaining gates we have to add the factor $(s - 1)! = s/s!$.

(2) Use $n \leq s$ and Stirling’s formula to conclude from $\log C(n, s)$ that a small fraction of circuits isn’t computable with at most $s := 2^n/n$ gates.

$$\log C(n, s) \leq \left(\frac{2^n}{n} + 1\right) (n - \log n) + (6 + \log e) \frac{2^n}{n} = 2^n - (1 - o(1)) \frac{2^n \log n}{n}$$

\implies a fraction of $2^{-(1-o(1)) \frac{2^n \log n}{n}}$ of the overall $|B_n| = 2^{2^n}$ functions can **not** be computed within the given amount of gates.

□

1.2 Formula Complexity

- A formula forms a binary tree. Therefore: #gates+1 = #leaves
- Let \mathcal{F} be a formula. Define $L(\mathcal{F}) = \#\text{leaves of } \mathcal{F}$ and $D(\mathcal{F}) = \text{depth of } \mathcal{F}$.
Important: Don’t confuse $L(\mathcal{F})$ and $L_\Omega(\mathcal{F})$!

Proposition 1.2.1. *Let f be a boolean function. If f can be computed by a circuit of depth d , then it can also be computed by a formula of depth d , i.e. the depth complexity of circuits and formulae are the same.*

Proof. This fact is easy to see. Just pick an arbitrary boolean function as well as an associated circuit. Let g be a gate in the circuit with fan-out $n > 1$. By copying the gate $n - 1$ times we gain a circuit consisting only of gates of fan-out at most 1 – which is a formula. □

Proposition 1.2.2. *Let Ω_1, Ω_2 be complete bases. Then $D_{\Omega_1}(f) = \Theta(D_{\Omega_2}(f))$.*

Proof. Intuition: By exchanging the bases we have to express some gates by constructs expressed in the new base. Since these newly expressed nodes are of finite size we get just a constant factor in front of the old circuit depth. □

Theorem 1.2.3. *For any complete basis Ω , $D_\Omega(f) = \Theta(\log L_\Omega(f))$ holds.*

Proof. (1) A formula forms a binary tree. We gain $L_\Omega(f) \leq 2^{\mathcal{O}(D_\Omega(f))}$, i.e. $\Omega(\log L_\Omega(f)) = D_\Omega(f)$.

(2) Now, it remains to proof that $D_\Omega(f) = \mathcal{O}(\log L_\Omega(f))$ holds.

Proposition 1.2.4. *For all binary trees T and $1 < m < n$ exist a tree-node x , such that $\#\text{leaves}(T_x) \geq m$, but $\#\text{leaves}(T_{x_1}), \#\text{leaves}(T_{x_2}) < m$ (x_i denotes a child of x in T).*

That's easy to see. Set $m := \frac{n}{3}$. Proposition 1.2.4 yields the existence of a tree-node x , such that $\#\text{leaves}(T_x) \geq m$. Furthermore, it implies

$$\begin{aligned} \#\text{leaves}(T_x) &= \#\text{leaves}(T_{x_1}) + \#\text{leaves}(T_{x_2}) < 2 \cdot m = \frac{2n}{3} \\ \implies \frac{n}{3} &\leq \#\text{leaves}(T_x) < \frac{2n}{3}. \end{aligned}$$

'Cause T is a binary tree node x has exactly one parent. By removing the connection between x and $\text{parent}(x)$ split T in two subtrees: T_1 , containing everything except T_x , and $T_2 := T_x$. By treating the binary trees as formulas (what they, in fact, are) then the correlation

$$T(\text{input}) = T_1(T_2(\text{input})) = (T_1(0) \wedge \overline{T_2(\text{input})}) \vee (T_1(1) \wedge T_2(\text{input})), \quad \text{input} \in \{0, 1\}^n \quad (1.2)$$

becomes clear. Denote by $D(k)$ the maximal depth complexity of any formula with $L(f) = k$. Conclude:

$$D(n) \leq D(2n/3) + 2 \leq \log_{3/2} n \approx 3.42 \log_2 n.$$

Remark: The “+2” term has been added because our selection construction (1.2) adds one layer for \wedge as well as another for \vee . □

Corollary. For any two complete bases Ω_1, Ω_2 , $L_{\Omega_1}(f) = (L_{\Omega_2}(f))^{\mathcal{O}(1)}$ holds.

Example 1.2.1. Show: $L_{U_2}(\text{Parity}_n) = \mathcal{O}(n^2)$.

Definition 1.2.1 (Partial assignment). Let $f : \{0, 1\}^n \rightarrow \{0, 1\}$ be a boolean function. A partial assignment is a function $\rho : \{1, 2, \dots, n\} \rightarrow \{*, 0, 1\}$ where $*$ denotes a variable, i.e. no assignment. We denote by f_ρ the resulting restriction of f .
With $R_m^n := \{\rho : \{1, 2, \dots, n\} \rightarrow \{*, 0, 1\} : |\rho^{-1}(*)| = m\}$ we denote the set of all partial assignments leaving exactly m out of n variables unassigned.

Lemma 1.2.5. Let $\rho \in R_{n-1}^n$ be a random partial assignment for a boolean function f . Then: $E(L(f_\rho)) \leq (1 - \frac{1}{n})L(f)$.

Proof. Denote by l_i the number of literals corresponding to variable x_i . Obviously, $\sum_{i=1}^n l_i = L(f)$ for $f \in B_n$. Let $\rho \in R_{n-1}^n$ be a random partial assignment, chosen randomly. Hence:

$$E(L(f_\rho)) = \sum_{i=1}^n l_i \cdot \underbrace{\Pr[\rho(x_i) = *]}_{=\varepsilon = \frac{n-1}{n}} \leq \frac{n-1}{n} \sum_{i=1}^n l_i = (1 - 1/n)L(f).$$

□

Lemma 1.2.6. Let $\rho \in R_{n-1}^n$ be a random partial assignment for a boolean function $f \in B_n$. If $L(f) = 1$, then $E(L(f_\rho)) = (1 - 1/n)L(f)$.

Proof. Let w.l.o.g. x_n be the only leaf. Then:

$$\begin{aligned} E(L(f_\rho)) &= \frac{1}{2n} \underbrace{\sum_{i=1}^n (L(f_{x_i=0}) + L(f_{x_i=1}))}_{\text{standard form}} \\ &= \frac{1}{2n} \underbrace{\sum_{i=1}^{n-1} (L(f_{x_i=0}) + L(f_{x_i=1}))}_{=2(n-1)L(f)} + \frac{1}{2n} \underbrace{(L(f_{x_n=0}) + L(f_{x_n=1}))}_{=0} \\ &= \frac{1}{2n} \cdot 2(n-1)L(f) = (1 - 1/n)L(f). \end{aligned}$$

□

Lemma 1.2.7. *Let $\rho \in R_{n-1}^n$ be a random partial assignment for a boolean function f . If $L(f) > 1$, then: $E(L_{U_2}(f_\rho)) \leq (1 - \frac{1.5}{n})L_{U_2}(f)$.*

Proof. Let x_i be a leaf and denote by l_i the number of occurrences of x_i .

$$\begin{aligned} \text{parent}(x_i) \hat{=} \wedge & \longrightarrow \begin{cases} L(f_{x_i=0}) \leq L(f) - 2l_i, \\ L(f_{x_i=1}) \leq L(f) - 1l_i \end{cases} \\ \text{parent}(x_i) \hat{=} \vee & \longrightarrow \begin{cases} L(f_{x_i=0}) \leq L(f) - 1l_i, \\ L(f_{x_i=1}) \leq L(f) - 2l_i \end{cases} \end{aligned}$$

(This only holds 'cause x_i 's sibling subtree isn't empty. Therefore by removing $\text{parent}(x_i)$ we'll remove at least another leaf as well.) Independent of $\text{parent}(x_i)$'s gate-type we'll save at least $3l_i$ many leaves. Hence:

$$E(L(f_\rho)) = \frac{1}{\#\text{summands}} \sum_{1 \leq i \leq n} \underbrace{L(f_{x_i=0}) + L(f_{x_i=1})}_{\leq L(f) - 3l_i} \leq \frac{2n}{2n} L(f) - \frac{3}{2n} \sum_{1 \leq i \leq n} l_i = L(f) \left(1 - \frac{3}{2n}\right).$$

□

Lemma 1.2.7 has told us how to bound the **expected formula complexity** by using random partial assignments. But how to get rid of the nasty condition $L(f) > 1$? The following lemma tells us how to do that (it's more a technical rewriting of the original statement).

Lemma 1.2.8. *Let $\rho \in R_{n-1}^n$ be a random partial assignment for a boolean function f . Then: $E(L_{U_2}(f_\rho) - \frac{1}{3}) \leq (1 - \frac{1.5}{n})(L_{U_2}(f) - \frac{1}{3})$.*

Proof. (1) $L(f) > 1$: Use Lemma 1.2.7.

(2) $L(f) = 1$:

$$\left(1 - \frac{1.5}{n}\right) \left(1 - \frac{1}{3}\right) = \frac{2}{3} - \frac{1}{n} \geq \frac{n-1}{n} E(L(f)) - \frac{1}{3} = \frac{2}{3} - \frac{1}{n}$$

(3) $L(f) = 0$: LHS = $-\frac{1}{3}$, RHS $> -\frac{1}{3}$.

□

Theorem 1.2.9 (Subbotovskaya). *Let $\rho \in R_m^n$ be a random partial assignment for a boolean function f . Then $E(L_{U_2}(f_\rho)) \leq (\frac{m}{n})^{1.5} (L_{U_2}(f) - \frac{1}{3})$.*

Proof. First we observe that a random partial assignment ρ can be written as a sequence $\rho_n, \rho_{n-1}, \dots, \rho_m$ with $\rho_i \in R_i^n$ for $m \leq i \leq n$ s.t. $f = f_{\rho_n}, f_{\rho_{n-1}}, \dots, f_{\rho_m} = f_\rho$. This is possible by choosing the ρ_i random and independent. As an abbreviation we denote f_{ρ_i} by f_i . The theorem can now be shown by induction, using the result of lemma 1.2.8:

$$\begin{aligned} E(L_{U_2}(f_m) - 1/3) & \leq \left(1 - \frac{1.5}{m+1}\right) \underbrace{(L_{U_2}(f_{m+1}) - 1/3)}_{=E(L_{U_2}(f_{m+1}) - 1/3)} \\ & \leq \left(1 - \frac{1.5}{m+1}\right) \left(1 - \frac{1.5}{m+2}\right) E(L_{U_2}(f_{m+2}) - 1/3) \\ & \dots \\ & \leq \underbrace{\prod_{i=m+1}^n \left(1 - \frac{1.5}{i}\right)}_{\leq (\frac{m}{n})^{1.5}} E(L_{U_2}(f_n) - 1/3) \end{aligned}$$

□

Example 1.2.2. $L_{U_2}(\text{XOR}_n) = \Omega(n^{1.5})$. Clear: Let $m = 1$ and choose arbitrary at random a partial assignment $\rho \in R_1^n$. Then: $L_{U_2}((\text{XOR}_n)_\rho) = 1$.

Proposition 1.2.10. $L_{U_2}(\text{XOR}_n) = \mathcal{O}(n^2)$.

Proof. \rightarrow Proposition 1.2.11. □

Another lower bound result by applying the theorem of Subbotovskaya.

Definition 1.2.2 (Andreev function). The Andreev function is somehow the formalization of indirect addressing. Formally: $\text{An} : \{0, 1\}^{2n} \rightarrow \{0, 1\}$, $(x, y) \mapsto y_i$ with $x, y \in \{0, 1\}^n$ and

$$(x, y) = (x_1, \dots, x_{\log n}, y) \text{ with } x_1, \dots, x_{\log n} \in \{0, 1\}^{n/\log n}.$$

$\text{An}(x, y) = y_i$ results from interpreting $(\text{XOR}_{\log n}(x_1), \dots, \text{XOR}_{\log n}(x_{\log n}))_{\text{binary}} = i_{\text{decimal}}$.

Proposition 1.2.11. $L_{U_2}(\text{An}) = \mathcal{O}(n^3)$.

Proof. Denote by $g_y : \{0, 1\}^{\log(n)} \rightarrow \{0, 1\}$ the formula describing the “ $\log(n)$ to 1” multiplexer used in An . As we know from Shannon’s theorem there exists another formula \tilde{g}_y (minimizing the gates used by g_y) with $L_{B_2}(\tilde{g}_y) \geq \frac{2^{\log(n)}}{\log(\log(n))}$. With $L_{U_2}(f) \in \Theta(L_{B_2}(f))$ (we only have to replace each node by a constant number of gates) for $f \in B_n$ we get $L_{U_2}(\tilde{g}_y) \in \mathcal{O}(n)$.

Now let’s move the focus to the parity operations An has to perform. There are $\log(n)$ many $\frac{n}{\log(n)}$ -times parity functions. We assume that we have a direct access to both literals of each variable (the original and the negated version). In a formula performing a k -times parity operation it suffices to involve at most each pair of literals, hence $(\frac{2n}{\log(n)})^2$ for each parity function and immediately following $\log(n) \cdot (\frac{2n}{\log(n)})^2 \in \mathcal{O}(n^2)$ for the whole block of parity operations.

Since we are talking about formulas (hence: binary trees) we know the number of gates (=nodes) in the tree: There are

$$2^{\log((cn)^2)+1} - 1 < 2 \cdot c^2 \cdot n^2 = c' \cdot n^2 \in \mathcal{O}(n^2)$$

gates needed to calculate the parity functions.

Let’s combine our results. Since we are talking about formulas and $y \in \{0, 1\}^n$ we have to copy the whole block performing the parity operations n times to use the results. We conclude $L_{U_2}(\text{An}) \in \mathcal{O}(n^3)$. □

Theorem 1.2.12.

$$L_{U_2}(\text{An}) = \Omega(n^{2.5-o(1)}) = \Omega\left(\frac{n^{2.5}}{\log^{1.5} n \cdot \log^{2.5} \log n}\right).$$

Proof. (1) Show lower bound for An^* , then use $L_{U_2}(\text{An}) \geq L_{U_2}(\text{An}^*)$.

(2) With lemma 1.2.13 we observe the following fact: Since we leave at least one variable per row unassigned we are still able to produce each possible input for y^* , representing the truth-table for the chosen $f^* \in B_{\log n}$.

(3) Use Shannon’s observation as well as lemma 1.2.13 & 1.2.14 to show that

$$E(L_{U_2}(\text{An}^*)) \geq \frac{1}{2} \cdot \frac{n}{\log \log n}.$$

By Shannon we already know that there exists a function $f \in B_{\log n}$ such that $L_{U_2}(f) \geq L_{B_2}(f) \geq n/\log \log n$. By lemma X & Y,

$$E(L_{U_2}(\text{An}^*)) \geq E_{\rho \in R_m^n}(L_{U_2}(\text{An}_\rho^*)) \geq \frac{1}{2} \cdot \frac{n}{\log \log n}.$$

(4) Finally use the result of Subbotovskaya.

$$E_{\rho \in R_m^n}(L_{U_2}(\text{An}_\rho^*)) \leq \left(\frac{m}{n}\right)^{1.5} \left(L_{U_2}(\text{An}^*) - \frac{1}{3}\right),$$

hence (with $m := 2 \log n \log \log n$)

$$\begin{aligned} L_{U_2}(\text{An}) &\geq L_{U_2}(\text{An}^*) \geq E_{\rho \in R_m^n}(L_{U_2}(\text{An}_\rho^*)) \cdot \left(\frac{m}{n}\right)^{-1.5} + \frac{1}{3} \\ &\geq \frac{1}{2} \cdot \frac{n}{\log \log n} \cdot \left(\frac{n}{2 \log n \log \log n}\right)^{1.5} = \Omega\left(\frac{n^{2.5}}{\log^{1.5} n \cdot \log^{2.5} \log n}\right). \end{aligned}$$

□

Lemma 1.2.13.

$$\forall \rho \in R_{2 \log n \log \log n}^n : \Pr \left[\bigwedge_{i=1}^{\log n} |\rho|_{x_i}^{-1}(\ast) \geq 1 \right] \geq \frac{1}{2}$$

Proof. (1) Show the lemma by proving the converse probability:

$$\Pr \left[\bigvee_{i=1}^{\log n} |\rho|_{x_i}^{-1}(\ast) = 0 \right] < \frac{1}{2}.$$

(2) First, assign $\ell \in \{0, 1, \ast\}$ independent and at random by ρ to each variable. Show for an arbitrary fixed free variable that the probability of **not** appearing in the i -th row is $1 - 1/\log n$.

$$\begin{aligned} &\Pr[\text{ a fixed free variable does **not** appear in the } i\text{-th row }] \\ &= 1 - \Pr[\text{ a fixed free variable does appear in the } i\text{-th row }] = 1 - \frac{1}{\log n}. \end{aligned}$$

(3) Show: $\Pr[\text{ none of the } m \text{ free variables appear in the } i\text{-th row }] < 1/\log^2 n$.

$$\begin{aligned} &\Pr[\text{ none of the } m \text{ free variables appear in the } i\text{-th row }] \\ &= \prod_{k=1}^m \underbrace{\Pr[\text{ the } k\text{-th free variable does not appear in the } i\text{-th row }]}_{=1 - \frac{1}{\log n}} \\ &= \left(1 - \frac{1}{\log n}\right)^m < \frac{2^{\log n}}{2^m} = \frac{2^{\log n}}{2^{2 \log n \log \log n}} = \frac{1}{\log^2 n}. \end{aligned}$$

(4) Conclude: $\Pr[\text{ there is a row containing no free variable }] < 1/2$ for $n \geq 5$.

$$\begin{aligned} &\Pr[\text{ there is a row containing no free variable }] \\ &= \Pr \left[\bigvee_{i=1}^{\log n} \text{ none of the } m \text{ free variables appear in the } i\text{-th row } \right] \\ &= \sum_{i=1}^{\log n} \Pr[\text{ none of the } m \text{ free variables appear in the } i\text{-th row }] \\ &= \log n \cdot \frac{1}{\log^2 n} = \frac{1}{\log n} < \frac{1}{2} \text{ for } n \geq 5. \end{aligned}$$

□

Lemma 1.2.14.

$$\forall \rho \in R_{\log n}^n : \left\{ \left(\bigwedge_{i=1}^{\log n} |\rho|_{x_i}^{-1}(\ast) \geq 1 \right) \text{ is true} \implies L_{U_2}(An_\rho^\ast) \geq \frac{n}{\log \log n} \right\}$$

Proof. We've chosen $f^\ast \in B_{\log n}$ such that $L_{U_2}(f^\ast) \geq n/\log \log n$. Now assume that a random partial assignment $\rho \in R_{\log n}^n$ leaves at least one variable per row unassigned. We immediately see from the construction of the Andreev function that we still can produce $2^{\log n}$ different value-combinations with $\log n$ blocks (because of the XOR). Hence, each possible input for f^\ast can be produced and since $L_{U_2}(f^\ast) \geq n/\log \log n$, it follows $L_{U_2}(An_\rho^\ast) = L_{U_2}(An_\rho(y^\ast)) \geq L_{U_2}(f^\ast) \geq n/\log \log n$. \square

Theorem 1.2.15. *There is $f \in B_n^1$ with $L_{U_2}(f) \geq \frac{2^n}{\log n}$.*

1.3 Monotone Circuits

Definition 1.3.1 (Monotone functions). A function $f \in B_n$ is called monotone if for all $x, y \in \{0, 1\}^n$ with $x \leq y$, $f(x) \leq f(y)$ holds. Denote by $MB_n := \{f \in B_n \mid f \text{ is monotone}\}$ the set of all monotone functions in n variables.

Definition 1.3.2 (Threshold-functions). Set $T_n^k(x) = 1 \iff \sum_{i=1}^n x_i \geq k$. With $k = n/2$ we get the **Majority**-function.

Example 1.3.1 (non-monotonic functions). $E_n^k(x) = 1 \iff \sum_{i=1}^n x_i = k$. With $k = 1$ we get the **Parity**-function.

Theorem 1.3.1. *$f \in B_n$ is a monotone function iff it is computable by a $\{\wedge, \vee\}$ -circuit.*

Proof. We prove the proposition by showing both directions.

(1) Let f, f_0, f_1 be monotone functions with $f_0 = f|_{x_n=0}, f_1 = f|_{x_n=1}$.

Proposition 1.3.2. $f_0 \leq f_1$.

f monotone $\implies f_0(x_1, \dots, x_{n-1}) = f(x_1, \dots, x_{n-1}, 0) \leq f(x_1, \dots, x_{n-1}, 1) = f_1(x_1, \dots, x_{n-1})$. We conclude: $f_0 \leq f_1$.

Obviously we can write f in the form $f = (x_n \wedge f_1) \vee (\neg x_n \wedge f_0)$. With the result from Proposition 1.3.2 we observe the relation

$$f = (f_0 \vee x_n) \wedge \underbrace{(f_1 \vee f_0)}_{=f_1} = (f_1 \wedge x_n) \vee \underbrace{(f_1 \wedge f_0)}_{=f_0}. \tag{1.3}$$

By applying the two possible values for x_n we easily observe the correctness:

$$\begin{aligned} x_n = 0 &\implies (f_0 \vee 0) \wedge f_1 = f_0 \wedge f_1 = f_0 \\ x_n = 1 &\implies (f_1 \wedge 1) \vee f_0 = f_1 \vee f_0 = f_1 \end{aligned}$$

Therefore the representation of f as a circuit over $\{\wedge, \vee\}$ follows by the result of (1.3). By induction we see that this representation holds for each $f \in B_n$.

(2) Let $f \in B_n$ be a function computable by a $\{\wedge, \vee\}$ -circuit and $x, y \in \{0, 1\}^n$ with $x \leq y$. To prove: f is monotonic. We show this by induction over the length l of the circuit f .

$l = 1$: We are just looking at one literal, therefore we get the identity function which is obviously a monotone one.

$l = 2$: Let $f_l \in B_l = B_2$ of the form $f_l(a, b) = a \wedge b$ (the same argument holds for \vee). With $x \leq y$ we get immediately that f_l is monotone.

$l > 2$: Let f_i, f_j be circuits over $\{\wedge, \vee\}$ with $1 \leq i, j < l, i + j = l$. By induction hypothesis we know that f_i, f_j are monotone functions. Same argument as above: Let $f_l := f_i \wedge f_j$ (the same holds for \vee) which is obviously a monotone function.

We get that f really is a monotone function.

□

Theorem 1.3.3. For almost all $f \in M_n$ the following bounds hold.

$$(1) C_{M_2}(f) = \Omega\left(\frac{2^n}{n^{1.5}}\right)$$

$$(2) L_{M_2}(f) = \Omega\left(\frac{2^n}{n^{0.5} \log n}\right)$$

$$(3) D(f) = \Omega\left(n - \frac{1}{2} \log n - \log \log n + c_2 \log n\right)$$

Theorem 1.3.4. Each monotone function $f \in M_n$ can be computed by a monotone circuit of size $\mathcal{O}\left(\frac{2^n}{n^{1.5}} \log n\right)$.

Combining these two results we see:

$$\Omega\left(\frac{2^n}{n^{1.5}}\right) \leq C_{M_2}(f) \leq \mathcal{O}\left(\frac{2^n}{n^{1.5}} \log n\right)$$

holds for almost all $f \in M_n$.

Example 1.3.2. $D_{M_2}(\text{Majority}_n) = \mathcal{O}(\log^2 n)$

(See: Lecture notes, T_n^k -construction)

Theorem 1.3.5 (Valiant). $D_{M_2}(\text{Majority}_n) \leq 5.3 \log n$.

1.4 Unbounded Fan-In Circuits

Theorem 1.4.1.

$$C_{M_2}(\text{CLIQUE}_3(n)) = \Omega\left(\frac{n^3}{\log^4 n}\right).$$

Proof. (1) Use the method of approximation and show that each approximation of $\text{CLIQUE}_3(n)$ is bad, i.e. each gate of an approx. for $\text{CLIQUE}_3(n)$ makes just a little error, but the overall error appears to be huge.

(2) Lemma 1.4.3 with $p \approx 3 \log n$ and $m \approx 18 \log^2 n$ implies the theorem. □

Remark. $(c_{i_1} \sqcap c_{i_2})(G) \leq (c_{i_1} \wedge c_{i_2})(G)$, $(c_{i_1} \sqcup c_{i_2})(G) \geq (c_{i_1} \vee c_{i_2})(G)$.

Lemma 1.4.2. For all $f : \{0, 1\}^n \rightarrow \{0, 1\}$ holds:

$$[f_t(G) = 1 \wedge c_t(G) = 0] \implies [it \text{ exists a gate } g \text{ such that } (c_{i_1} \sqcap c_{i_2})(G) < (c_{i_1} \wedge c_{i_2})(G)]$$

Analogous for \sqcup/\vee .

Lemma 1.4.3. *Each monotone circuit approximating $\text{CLIQUE}_3(n)$ contains at least either $\binom{n}{3}/m^2$ AND-gates or 2^{p-1} OR-gates.*

Proof. (2) Error also on $c_t \Rightarrow$ Error on at least 2^{n-1} negative testgraphs \Rightarrow prob. of hitting a negative test graph fulfilling the inequality is at least $2^{-(n-p)} \Rightarrow$ at least $2^{n-1}/2^{-(n-p)} = 2^{p-1}$ errors occur and since we had only considered OR-gates this will be the lower bound of errors appearing at them. \square

Lemma 1.4.4.

$$\left(\bigvee_{\{i,j\} \in A} x_{i,j} \right) \sqcap \left(\bigvee_{\{i,j\} \in B} x_{i,j} \right) < \left(\bigvee_{\{i,j\} \in A} x_{i,j} \right) \wedge \left(\bigvee_{\{i,j\} \in B} x_{i,j} \right)$$

for at most m^2 positive testgraphs.

Proof. Assume that the inequality holds. Clearly, LHS = 0, RHS = 1. Because of RHS = 1 both A and B contain at least one edge. Observe that $A \cap B = \emptyset$ since LHS = 0. Hence the (at least) two edges are part of a triangle if G is a positive test graph. Furthermore, $|A|, |B| \leq m$. Let's assume w.l.o.g. that $|A| > m, |B| \leq m$. Then $\bigvee_{\{i,j\} \in A} x_{i,j}$ will evaluate to be constantly 1. But right then we would observe $\bigvee_{\{i,j\} \in B} x_{i,j} < \bigvee_{\{i,j\} \in B} x_{i,j}$ which is impossible. With that in mind we conclude that only $|A| \cdot |B| \leq m^2$ many positive test graphs will satisfy the proposed inequality. \square

Lemma 1.4.5.

$$\left(\bigvee_{\{i,j\} \in A} x_{i,j} \right) \vee \left(\bigvee_{\{i,j\} \in B} x_{i,j} \right) < \left(\bigvee_{\{i,j\} \in A} x_{i,j} \right) \sqcup \left(\bigvee_{\{i,j\} \in B} x_{i,j} \right)$$

holds for at most 2^{n-p} negative testgraphs.

Proof. Assume the inequality holds, i.e. LHS = 0, RHS = 1. The latter one being constantly 1 implies $|A \cup B| > m$. Show: For each negative test graphs G , chosen arbitrary at random, holds

$$\underbrace{\Pr[G \text{ satisfies the inequality above }]}_{=\Pr[\forall \text{ edge}(i,j) \in A \cup B : \text{color}(i) = \text{color}(j)]} \leq 2^{-p}.$$

Easy to see:

$$\begin{aligned} \text{Inequality holds} &\iff x_{i,j} = 0 \forall \{i,j\} \in A \cup B \\ &\iff \forall \{i,j\} \in A \cup B : \nexists \text{ edge}(i,j) \in E \\ &\iff \text{color}(i) = \text{color}(j) \forall \{i,j\} \in A \cup B \end{aligned}$$

Consider two cases:

(1) G contains a matching of size $\tilde{p} \geq p$. To fulfill the inequality we have to show that

$$\Pr\left[\bigwedge_{k=1}^{\tilde{p}} \text{color}(i_k) = \text{color}(j_k) \right] \leq 2^{-\tilde{p}} \leq 2^{-p}.$$

But since the probability that an edge contained in the matching connects to like-colored vertices is independent of the coloring of any other vertex covered by the matching we immediately observe our bound.

(2) G contains a star of size $\tilde{p} \geq p$. Same argument as in the first case.

It remains to show that at least one of the two cases stated above is true. \rightarrow see lemma 1.4.6. \square

Lemma 1.4.6. *Each set of **more** than $m = 2(p-1)^2$ edges either contains a matching of size $\geq p$ or a star of size $\geq p$.*

Proof. If the given set contains a matching of size $\geq p$ we're done. Therefore assume that the maximal matching is of size $\leq p-1$. Then at most $2(p-1)$ vertices are covered by the matching. On average a vertex covered by the matching touches more than $m/2(p-1) = p-1$ edges in the given set. Because this is the average value, there will be a vertex which touches more than $p-1$ (\equiv at least p) edges. Hence this vertex will be the center of a star of size at least p . \square

Lemma 1.4.7. *Consider $c = \bigvee_{\{i,j\} \in A} x_{i,j}$. Then c is either 0 on all positive test graphs or 1 on at least half of the negative test graphs.*

The proof is nearly trivial. We need this lemma to show that each approximator itself is very bad, but the connection of the approximators is reasonably good.

1.5 A lower bound for Parity_n

Lemma 1.5.1. *Für alle Wk'verteilungen ω auf $\{0,1\}^n$ und alle $\delta > 0$ gibt es ein Polynom $p \in GF_3[X_1, \dots, X_m]$ mit $\deg(p) \geq 2\lceil \log(1/\delta) \rceil$, s.d.*

$$\Pr_{x \in \{0,1\}^n} [OR(x) \neq p(x)] \leq \delta.$$

Proof. (1) For $S \subseteq \{1, \dots, n\}$ define $f_S(x) := (\sum_{i \in S} x_i)^2$.

(2) Halbseitiger Fehler: $OR(x) = 0 \implies f_S(x) = 0, OR(x) = 1 \implies \Pr_S[f_S(x) = 1] \leq 1/2$. Wähle $S \subseteq \{1, \dots, n\}$ zufällig gleichverteilt. Dann:

$$\begin{aligned} \Pr_s[f_S(x) = 1] &= \frac{1}{2^n} \sum_{S \subseteq \{1, \dots, n\}} f_S(x) = \frac{1}{2^n} \sum_{S' \subseteq \{1, \dots, n\} \setminus \{i_0\}} (f_{S'}(x) + f_{S' \cup \{i_0\}}(x)) \\ &\geq \frac{1}{2^n} \sum_{S' \subseteq \{1, \dots, n\} \setminus \{i_0\}} 1 = \frac{1}{2^n} \cdot 2^{n-1} = \frac{1}{2}. \end{aligned}$$

(3) $f_{S_1, \dots, S_\ell}(x) := 1 - \prod_{i=1}^\ell (1 - f_{S_i}(x))$ mit $\ell := \lceil \log(1/\delta) \rceil$

(4) Zeige $\forall x \in \{0,1\}^n : \Pr[f_{S_1, \dots, S_\ell}(x) \neq OR(x)] \leq \delta$ durch *Wahrscheinlichkeitsverstärkung*.

Sei $OR(x) = 1$. Wähle zufällig gleichverteilt und unabhängig voneinander Mengen $S_1, \dots, S_\ell \subseteq \{1, \dots, n\}$.

$$\begin{aligned} \Pr[f_{S_1, \dots, S_\ell}(x) = 1] &= 1 - \Pr[\forall 1 \leq i \leq \ell : f_{S_i}(x) = 0] \\ &= 1 - \prod_{i=1}^\ell \Pr[f_{S_i}(x) = 0] \geq 1 - (1/2)^\ell \geq 1 - \delta \end{aligned} \tag{1.4}$$

(5) Zeige $\Pr_\omega[f_{S_1, \dots, S_\ell}(x) \neq OR(x)] \leq \delta$ für bel. Wk'vertteilung ω .

Sei $X_{x, S_1, \dots, S_\ell} = 1 : \iff f_{S_1, \dots, S_\ell}(x) \neq OR(x)$.

$$\begin{aligned} \sum_{S_1, \dots, S_\ell} \left(\sum_{x \in \{0,1\}^n} \omega(x) \cdot X_{x, S_1, \dots, S_\ell} \right) &= \sum_{x \in \{0,1\}^n} \left(\sum_{S_1, \dots, S_\ell} \omega(x) \cdot X_{x, S_1, \dots, S_\ell} \right) \\ &= \sum_{x \in \{0,1\}^n} \omega(x) \cdot \left(\sum_{S_1, \dots, S_\ell} X_{x, S_1, \dots, S_\ell} \right) \\ &\stackrel{(1.4)}{\leq} \underbrace{\sum_{x \in \{0,1\}^n} \omega(x)}_{=1} \cdot 2^{n\ell} \cdot \delta = 2^{n\ell} \cdot \delta \end{aligned}$$

$\sum_{k=1}^n \binom{n}{k} = 2^n$ und ℓ Kombinationen (da für jedes S_i) $\implies (2^n)^\ell$ mögliche Teilmengenkombinationen. Andere Möglichkeit der Veranschaulichung: Assoziiere eine konkrete Wahl $\mathcal{S} = (S_1, \dots, S_\ell)$ mit einem 0-1-Vektor $(b_{1,1}, \dots, b_{1,n}, b_{2,1}, \dots, b_{2,n}, \dots, b_{\ell,n})$ für den gilt: $b_{i,j} = 1 \iff j \in S_i$. \square

Lemma 1.5.2. Für alle Schaltkreise für $f \in B_n$ mit Tiefe d und #Gatter s und für alle $\varepsilon \geq 0$ existiert ein Polynom $p \in GF_3[X_1, \dots, X_n]$ mit $\deg(p) \geq (2 \lceil \log(s/\varepsilon) \rceil)^d$, s.d.

$$\Pr_{x \in \{0,1\}^n} [f(x) \neq p(x)] \leq \varepsilon.$$

Proof. (1) Setze $\delta = \varepsilon/s$ und bezeichne mit $p_{\text{AND}}, p_{\text{OR}}$ die Polynomapproximationen für AND/OR

(2) Ordne Schaltkreis Level für Level Approximationspolynome vom Grad $\leq D^i$ mit $D := 2 \lceil \log(s/\varepsilon) \rceil$ zu

(3) Betrachte, wie viele Fehler pro Level gemacht werden. Zeige durch Fallunterscheidung, dass die Fehler-Wk $\leq \delta$ auf Fehler-freier Menge $U \subseteq \{0,1\}^n$ nach oben durch δ bechränkt ist.

(4) Daher: Auf allen Bausteinen werden höchstens $s \cdot 2^n \cdot \delta = 2^n \cdot \varepsilon$ Fehler gemacht. \square

Lemma 1.5.3. Every multilinear polynomial $p \in GF_3[X_1, \dots, X_n]$ with $\deg(p) \leq \sqrt{n}$ satisfies $\Pr_{x \in \{0,1\}^n} [\text{Parity}_n(x) \neq p(x)] \geq 0.15$.

Proof. \rightarrow Lemma 1.5.4 \square

Define $\text{Parity}_n^*(x) := \prod_{i=1}^n x_i$ for $x \in \{-1,1\}^n$.

Lemma 1.5.4. Let $p : \{-1,1\}^n \rightarrow \{-1,1\}$, $p \in GF_3[X_1, \dots, X_n]$ be an arbitrary multilinear polynomial with $\deg(p) \leq \sqrt{n}$. Then $\Pr_{x \in \{-1,1\}^n} [\text{Parity}_n^*(x) = p(x)] \leq 0.85$.

Proof. (1) Look at the set of all x for whom $p(x) = \text{Parity}_n^*(x)$

$$\begin{aligned} \text{EXACT} &:= \{x \in \{-1,1\}^n \mid p(x) = \text{Parity}_n^*(x)\}, \\ F &:= \{f : \text{EXACT} \rightarrow \{-1,0,1\}\}. \end{aligned}$$

Hence: $p(x) = \text{Parity}_n^*(x) = \prod_{i=1}^n x_i$ for all $x \in \text{EXACT}$.

(2) Show that the degree of p can be reduced to be at most $\frac{1}{2}(n + \sqrt{n})$ Let S_i be an index map with $|S_i| > \frac{1}{2}(n + \sqrt{n})$. Then

$$\tilde{p}(x) := \prod_{j \in S_i} x_j = \prod_{j=1}^n x_j / \prod_{j \notin S_i} x_j = \prod_{j=1}^n x_j \cdot \prod_{j \notin S_i} x_j = p(x) \cdot \prod_{j \notin S_i} x_j$$

for all $x \in \text{EXACT}$. It follows:

$$\deg(\tilde{p}) = \deg(p) + (n - |S_i|) \leq \sqrt{n} + \left(n - \frac{1}{2}(n + \sqrt{n})\right) = \frac{1}{2}(n + \sqrt{n}).$$

(3) Conclude: $\#\text{EXACT} \leq \sum_{i=0}^{\frac{1}{2}(n+\sqrt{n})} \binom{n}{i} \leq 0.85 \cdot 2^n$

$$3^{\#\text{EXACT}} = \#F \leq \#\left\{p \in GF_3[X_1, \dots, X_n] \mid \deg(p) \leq \frac{1}{2}(n + \sqrt{n})\right\} = 3^{\sum_{i=0}^{\frac{1}{2}(n+\sqrt{n})} \binom{n}{i}} \leq 3^{0.85 \cdot 2^n}.$$

\square

Chapter 2

Computations over the Reals and over the Integers

Definition 2.0.1 (Subset Sum). $K_n := \{x \in \mathbb{R}^n \mid \exists \alpha \in \{0, 1\}^n \text{ with } \alpha x = 1\}$, $K = \bigcup_{n>0} K_n$.

Theorem 2.0.5 (MadH). For K_n exists a LDT of depth $\mathcal{O}(n^5 \log n)$.

Definition 2.0.2 (S-sets). Let's first introduce the notion of \mathcal{S} -functions. A function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ that can be computed by a straight line program of (the set of operations) \mathcal{S} is called a **\mathcal{S} -function**. Now consider a node v and denote by $c(v) \subseteq \mathbb{R}^n$ the set of inputs, arriving at node v . Further, let $\mathcal{M} := \{f_1, \dots, f_k\}$ be the set of \mathcal{S} -functions along all the paths for all $x \in c(v)$ to the node v . We'll call \mathcal{M} an **\mathcal{S} -set**.

Remark.

$L \subseteq \mathbb{R}^n$ can be recognized by an S-CT $\iff L$ is the union of finitely many S-sets.

Concrete:

$L \subseteq \mathbb{R}^n$ can be rec'd by a $\{+, -, *_c\}$ -CT $\iff L$ can be rec'd by an LDT
 $\iff L$ is the union of finitely many convex polytops

and

$L \subseteq \mathbb{R}^n$ can be rec'd by a $\{+, -, *\}$ -CT $\iff L$ can be rec'd by an ADT
 $\iff L$ can be rec'd by an ACT
 $\iff L$ is the union of finitely many semi-algebraic sets

Theorem 2.0.6. ED_n contains of at least $n!$ connected components, SD_n of at least $(n!)^2$ connected components.

Proof. (1) Consider an $x = (x_1, \dots, x_n) \in ED_n$ and let $\pi \in S_n \setminus \{\text{id}_n\}$ be an arbitrary permutation. Then $y := (x_{\pi(1)}, \dots, x_{\pi(n)})$ is also an element of ED_n . Furthermore, we know that there are i, j with $i < j$ such that $x_i < x_j$, but $y_i > y_j$. Thus, by the intermediate value theorem, every continuous path from x to y must contain a point z with $z_i = z_j$. Hence, x and y are lying in different connected components.

(2) Consider a $2n$ -tuple $(x; y) = (x_1, \dots, x_n, y_1, \dots, y_n) \in SD_n$. Since the sets $X = \{x_1, \dots, x_n\}, Y = \{y_1, \dots, y_n\}$ are disjoint subsets of them will satisfy a partial ordering $X_1 < Y_1 < X_2 < \dots < X_k < Y_k$. For an arbitrary k there are at most $(k!)^2$ different partial orders. The proposed number of connected components follows with $k = n$.

□

Remark. $L(n, k)$ can be written as a union of $(k + 1)^n - 1$ hyperplanes.

Proof.

$$L(n, k) = \bigcup_{\alpha \in \{0, 1, \dots, k\}^n \setminus \{\bar{0}\}} \{x \in \mathbb{R}^n \mid \alpha x - 1 = 0\} \implies \#L(n, k) = |\{0, 1, \dots, k\}^n \setminus \{\bar{0}\}| = (k + 1)^n - 1.$$

□

Corollary. Each LDT for ED_n or SD_n has depth $\Omega(n \log n)$.

Theorem 2.0.7. $\overline{L(n, k)} = \mathbb{R}^n \setminus L(n, k)$ contains of at least $(k + 1)^{\binom{n}{2}} \cdot 2^{-n}$ connected components.

Proof. **TODO**

□

Definition 2.0.3 (AP-languages).

$$L(\lambda, A, B) := B \cup \bigcup_{a \in A} \underbrace{\{a + \lambda x \mid x \in \mathbb{N}\}}_{=AP(a, \lambda)} \text{ for finite sets } A, B \subseteq \mathbb{N}$$

Theorem 2.0.8. $CC_1(\{+, -, \text{Div}\}) = \{AP\text{-languages}\}$

Theorem 2.0.9 (Dobkin/Lipton). $D_{LDT}(L) \geq \log_3(\#_1 L + \#_0 L)$.

Proof. #leaves $\leq 3^t$ and a leaves accept, r leaves reject. Obviously, #leaves = $a + r$.

Hence: $\#_0 L \leq r, \#_1 L \leq a \implies \#_0 L + \#_1 L \leq \#leaves \leq 3^t$.

□

Theorem 2.0.10 (Milnor). Let $X \subseteq \mathbb{R}^n$ be a semi-algebraic set comprised of m equalities and n inequalities, each of degree at most $d \geq 2$. Then X has at most $d(2d - 1)^{n+h-1}$ connected components

Corollary (Naive lower bound). Each ACT recognizing L has depth $\Omega(\min\{\sqrt{S}, \frac{S}{n}\})$ with $S := \log(\#_1 L + \#_0 L)$.

Proof. Consider an ACT of depth t for L . What we are searching for is a lower bound for t . So, because of depth t there are at most 3^t leaves. Furthermore the ACT computes polynomials of degree at most $d := 2^t$. The latter one can be easily seen by reconsidering the allowed operations:

$$y_0 := x_i \quad \rightarrow \quad y_1 := y_0^2 \quad \rightarrow \quad y_2 := y_1^2 \quad \rightarrow \cdots \rightarrow \quad y_t := y_{t-1}^2 = x_i^{2^t}$$

Milnor's result (Theorem 2.0.10) leads us to the following intermediate conclusion:

$$\begin{aligned} \#_1 L + \#_0 L &\leq 3^t \cdot 2^t (2 \cdot 2^t - 1)^{n+t-1} = 2^{\mathcal{O}(t(n+t))} \\ \implies \log(\#_1 L + \#_0 L) &\leq ct^2 + cnt. \end{aligned}$$

Depending on if t^2 or nt dominates the RHS, we can choose an appropriate c' such that either $c't^2 \geq ct^2 + cnt$ or $c'nt \geq ct^2 + cnt$. Solving the gained inequality for t yields the proposed lower bound:

$$\begin{aligned} \log(\#_1 L + \#_0 L) \leq c't^2 &\implies t = \Omega(\sqrt{\log(\#_1 L + \#_0 L)}), \\ \log(\#_1 L + \#_0 L) \leq c'nt &\implies t = \Omega(\log(\#_1 L + \#_0 L) / n). \end{aligned}$$

□

Theorem 2.0.11 (Ben-Or). Each ACT for $L \subseteq \mathbb{R}^n$ has depth $\Omega(\log(\#_1 L + \#_0 L) - n)$.

Proof. (1) Consider a leaf ℓ in an ACT for L and let (v_1, v_2, \dots, v_h) be a path through (in-)equalities from the tree's root to ℓ .

(2) Use the equalities and inequalities e_1, \dots, e_h on this path to construct a semi-algebraic set $\mathcal{U}(\ell) \subseteq \mathbb{R}^{n+h}$.

(3) Apply Milnor's theorem to get $\#_1\mathcal{U}(\ell) \leq 2 \cdot 3^{n+h-1}$.

(4) Project $\mathcal{U}(\ell)$ into the n -dimensional space \mathbb{R}^n and observe $\#_1c(\ell) \leq 2 \cdot 3^{n+h-1}$.

$\Pi(\mathcal{U}(\ell)) = c(\ell)$ since $x \in c(\ell) \iff \exists y \in \mathbb{R}^h : (x, y) \in \mathcal{U}(\ell)$. Hence: $\#_1c(\ell) \leq \#_1\mathcal{U}(\ell) \leq 2 \cdot 3^{n+h-1}$.

(5) Conclude: $\#_1L + \#_0L \leq 2^{h+1} \cdot 3^{n+h-1}$.

$$\begin{aligned} \#_1L + \#_0L &\leq 2^{h+1} \cdot 3^{n+h-1} = 6^h \cdot 2 \cdot 3^{n-1} \\ \iff \log_6 \left(\frac{\#_1L + \#_0L}{2 \cdot 3^{n-1}} \right) &\leq h \end{aligned}$$

□

Die Projektion $\Pi : \mathbb{R}^{n+m} \rightarrow \mathbb{R}^n$ erzeugt keine neuen Zusammenhangskomponenten, einfach einzusehen anhand folgender Argumentation: Sei \mathcal{C} eine bel. Zusammenhangskomponente und $a, b \in \mathcal{C}$ ebenfalls beliebig. Dann existiert eine kontinuierliche Funktion f , s.d. $f(0) = a, f(1) = b$. Sie beschreibt einen Pfad von a nach b in \mathcal{C} . Wir können nun f eindeutig in $n+m$ einzelne Komponentenfunktionen f_1, \dots, f_{n+m} zerlegen, s.d. $f(x) = (f_1(x_1), \dots, f_{n+m}(x_{n+m}))$. Jede Komponentenfunktion f_i beschreibt einen kontinuierlichen Pfad von a_i nach b_i in ihrer Dimension. Damit f einen Pfad im $n+m$ dimensionalen Raum beschreibt ist es daher notwendig, dass jede Komponentenfunktion einen kontinuierlichen Pfad beschreibt in ihrer Dimension. Damit erhält die Einschränkung auf die ersten n Komponentenfunktionen die Kontinuitätseigenschaft und $\Pi(\mathcal{C})$ ist in \mathbb{R}^n weiterhin zusammenhängend.