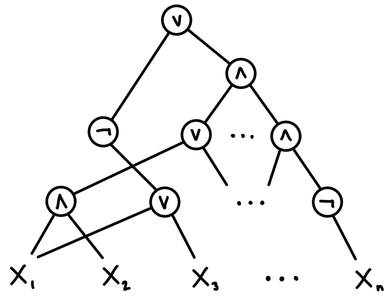
Monotone Circuits Struggle with Cliques

Styopa Zharkov

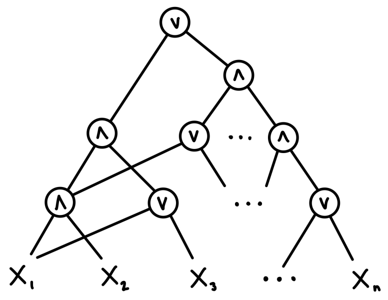
Introduction

Recall the circuit method of computation, where for each input length n we construct a circuit of OR, AND, and NOT gates on n bits of input. We measure the circuit complexity of a function by the smallest number of gates we need to compute the function using a circuit.



Here, we will look at a special subset of circuits called monotone circuits. We will prove a lowerbound on monotone circuits (i.e. monotone circuits are weak in some sense).

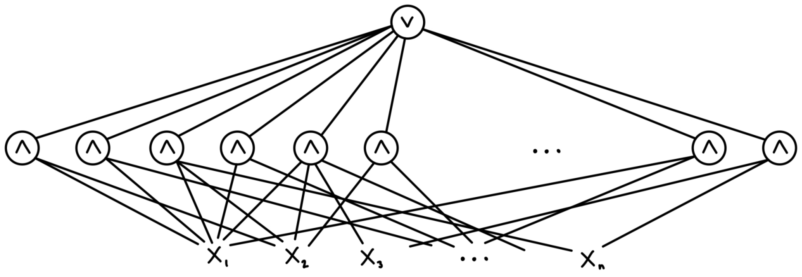
Definition 1: A circuit that has only OR and AND gates is called a *monotone circuit*. We will also restrict ourselves to a maximum fan-in of 2. ◁



Monotone circuits can't compute every function. For example, any circuit computing XOR needs a NOT gate. So, we introduce the following notation.

Definition 2: Let $x, y \in \{0,1\}^n$. We say $x \preceq y$ if every bit that's 1 in x is also 1 in y. A function $f : \{0,1\}^n \to \{0,1\}$ is *monotone* if $f(x) \leq f(y)$ when $x \preceq y$. ◁

Intuitively, this means that the value of f never decreases as we flip 0s to 1s in the input. In any monotone circuit, changing a 0 to a 1 in the input can never make any gate switch from 1 to 0, so the function it computes is monotone. For any monotone function, we can create a circuit that computes it by taking an OR of ANDs where each AND corresponds to an accepted input[1]. No extra inputs will be accepted because the function is monotone.



[1]To make the fan-in be 2, we can expand all AND and OR gates into several fan-in 2 AND and OR gates.

So, monotone functions are exactly the functions that monotone circuits can compute. However, the circuits we create for some functions in that way can be very large (where large means lots of gates). How small can we make these circuits if we try harder? We will approach this question by examining a specific monotone function (Clique) and show that there is no way to construct a small monotone circuit for it.

Definition 3: Let $\textsc{Clique}_{k,n} : \{0,1\}^{\binom{n}{2}} \to \{0,1\}$ be the function that takes as input an adjacency matrix for an n-vertex graph and outputs 1 if and only if the graph contains a k-clique (a fully connected subgraph of size k). ◁

Note that this function is monotone because adding an edge to a graph cannot destroy an existing clique. We must be careful when computing the complexity of this function in terms of input size because the input size is $\binom{n}{2}$ and not n. However, we know $\binom{n}{2} \leq n^2$ and we'll see that our bound is strong enough for this difference to not matter. We will prove that the monotone circuit complexity of this function is exponential in k (as long as k isn't too big in comparison to n).

Theorem 1: There exists an $\varepsilon > 0$ such that for large enough n and k where $k \leq n^{1/4}$, any monotone circuit computing $\textsc{Clique}_{k,n}$ has size at least $n^{\varepsilon\sqrt{k}}$. ◁

In other words, the monotone circuit complexity of $\textsc{Clique}_{k,n}$ is $n^{\Omega(\sqrt{k})}$. This was first proved by Razborov in 1985 and two years later by Alon and Boppana. Note that if we let $k = n^{1/4}$, then this bound becomes $n^{\Omega(n^{1/8})} \geq 2^{\Omega(n^{1/8})}$, so this bound is, in fact, exponential[2] in n.

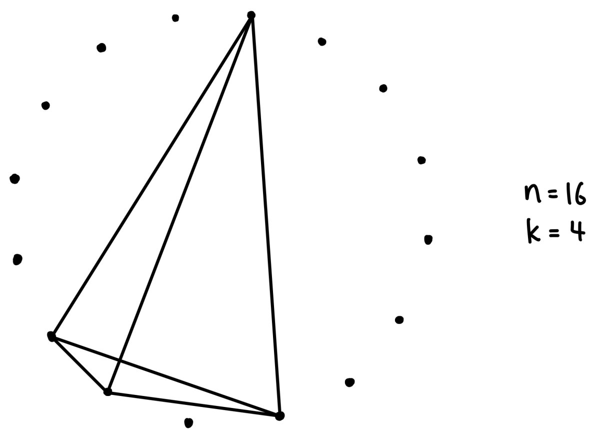
We can see that if we were able to prove a similar statement but for general circuits and not just monotone ones, then we have found a language in NP that isn't in P/poly. In fact, the original motivation for proving lower bounds on monotone circuits was a hope that we could show that any problem that's hard for monotone circuits is also hard for general circuits. If that were true, we would be able to prove that NP $\nsubseteq$ P/poly. Unfortunately this hope was crushed by the Tardos function, which is computable in polynomial time on normal circuits, but is exponentially hard for monotone circuits.

Yes, No, Indicators, and Approximators

Before we can approach the proof Theorem 1, we must define some tools we will use. First let's define the Yes and No type of graphs.

Definition 4: Let an n-vertex graph be called a $\text{Yes}_{k,n}$ graph if it has a k-clique somewhere and no other edges. Let the collection of Yes graphs be called $\mathcal{Y}_{k,n}$. ◁

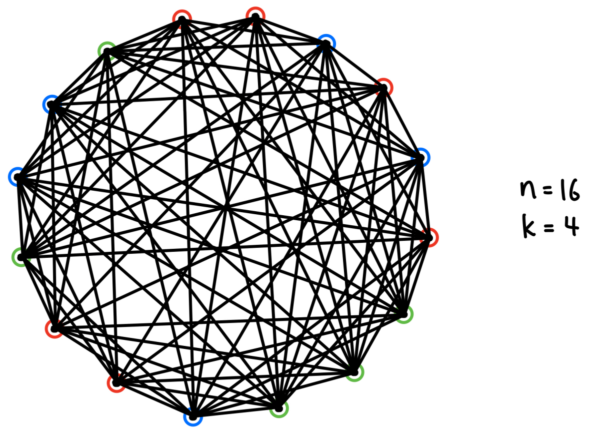
Intuitively, these are the sparsest graphs that still have a clique.



[2]Also note that any unbounded fan-in circuit can be turned into a fan-in of 2 circuit with only a linear expansion in size because the maximum fan-in of a circuit is no more than its size and it's possible to replace an unbounded gate of fan-in t with $t-1$ gates of fan-in 2. Since the bound we are proving is exponential, it also works for unbounded fan-in circuits.

Definition 5: Let an n-vertex graph be called a $\text{No}_{k,n}$ graph if it is formed by coloring the n vertices with $k-1$ colors and then connecting vertices of different colors. Although two different colorings can produce the same graph, we will consider them different to simplify our counting later. Let the collection of $\text{No}_{k,n}$ graphs along with their colorings be called $\mathcal{N}_{k,n}$. ◁

For any coloring and any group of k vertices, there are at least two that are the same color by the pigeonhole principle, which won't have an edge between them. So there are no k-cliques in $\text{No}_{k,n}$ graphs. Intuitively, there are many edges in most No graphs[3], so No graphs are very dense graphs without a clique.



Note that we will sometimes drop the k, n subscripts in our notation to keep our proofs clean.

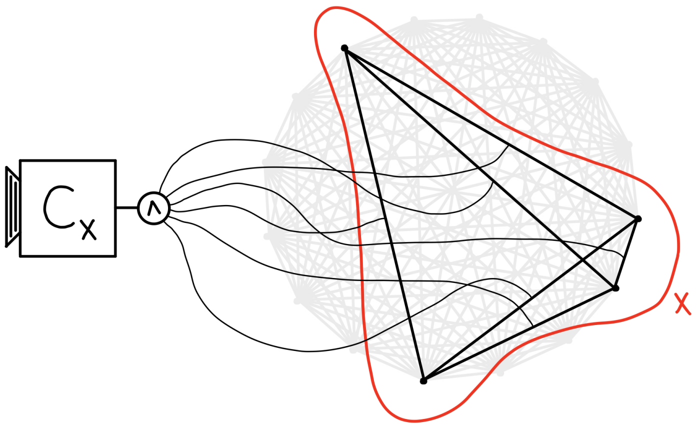
Next, we define a special kind of circuit.

Definition 6: Let X be a subset of $[n]$. The function that outputs 1 if and only if the input graph has a clique on the set X is called a *clique indicator* over X. We denote this function by C_X. ◁

The function C_X is just the monomial

$$C_X = \bigwedge_{i \neq j \in X} x_{ij}$$

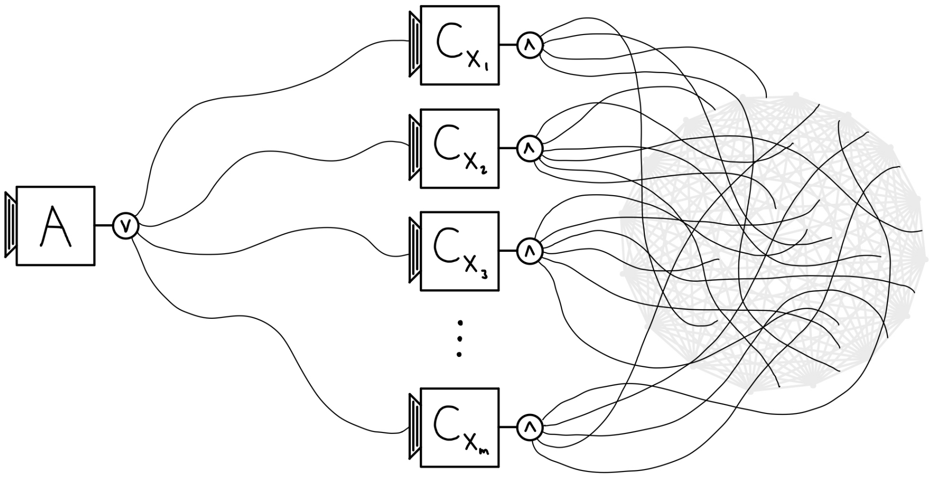
where x_{ij} is the bit representing the edge between vertex i and vertex j.



These indicators are the building blocks for another kind of circuit that is the true tool we are interested in.

[3]Unless there happens to be a lot of one color, which is unlikely.

Definition 7: An (m, ℓ)-*approximator* is an OR of at most m clique indicators, each of which is over a set with at most ℓ elements. ◁



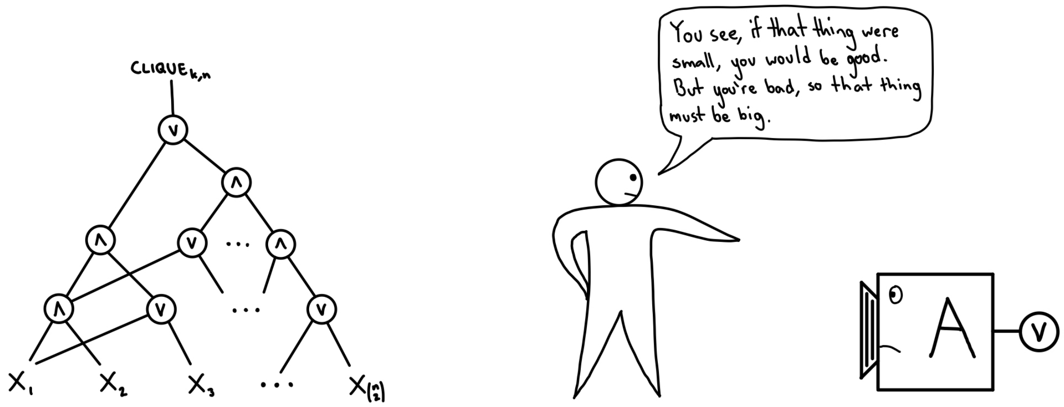
We can also write an (m, ℓ)-approximator A as

$$A = \bigvee_{t=1}^{r} C_{X_i} = \bigvee_{t=1}^{r} \bigwedge_{i \neq j \in X} x_{ij} \qquad (r \leq m, |X_i| \leq \ell).$$

As the name suggests, we will use approximators to approximate circuits. The parameters m and ℓ represent how large we allow the approximator to be. We will pick the specific values for these parameters later, but intuitively, we want to balance them. If they're too big or too small, then the approximator we can construct will be either too powerful or too weak to be useful.

Bad at Cliques, Good at Small Circuits

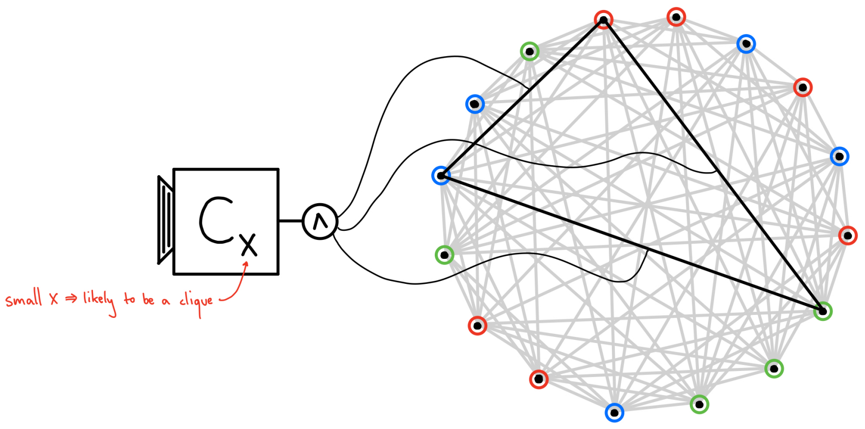
Razborov's idea is to carefully pick an m and ℓ big enough so that we can construct a good (m, ℓ) approximator for any small circuit, but also small enough so we can prove that any (m, ℓ)-approximator is bad at approximating $\textsc{Clique}_{k,n}$. Succeeding at this would imply that no small circuit can compute $\textsc{Clique}_{k,n}$, which is our end goal.


CLIQUE$_{k,n}$
X_1
X_2
X_3
$X_{\binom{n}{2}}$
You see, if that thing were small, you would be good. But you're bad, so that thing must be big.
A

We will restrict our attention on Yes and No graphs and prove these results on this subset of inputs specifically. We'll talk about how to construct the approximator for a small circuit computing $\textsc{Clique}$ in detail later, but the idea is to work from the bottom up. We start by constructing trivial (m, ℓ)-approximators for the input variables and then create approximators for every gate based on the approximators of the two gates it relies on.

Approximators are Bad at Identifying Cliques

First, let's explore how bad (m, ℓ)-approximators are at identifying cliques on Yes and No graphs. Intuitively, if ℓ is small compared to k, then the indicators in our approximator are small. Since $\text{No}_{k,n}$ graphs are dense, it's very likely that a randomly picked one will make an indicator return 1, so the approximator will wrongly return 1 as well. In fact, one small indicator is enough to make the approximator fail on many No graphs.



On the other hand, if ℓ is large compared to k and all of the indicators are over large sets, then it's likely that a random $\text{Yes}_{k,n}$ graph will be unrecognized by all of the indicators, so the approximator will wrongly return 0. So, approximators are just generally bad at identifying cliques.

When we construct our approximator for small circuits, we will set ℓ to be relatively small, so we don't need the second statement. So, we will only prove the case for small ℓ, but it is a good exercise to formalize the second statement as well.

Lemma 1: Every (m, ℓ)-approximator either rejects all graphs or wrongly accepts at least a $1 - \ell^2/(k-1)$ fraction of all $(k-1)^n$ graphs with colorings from $\mathcal{N}$. ◁

Proof (Lemma 1): If an (m, ℓ)-approximator accepts at least one graph, then it contains at least one clique indicator. Let's look at how one indicator C_X performs on No graph inputs.

A No graph and coloring are rejected if the subgraph on X isn't a clique. This is true if and only if two vertices in X are colored the same. There are $\binom{|X|}{2}$ pairs of vertices in X. For each pair, there are $(k-1)^{n-1}$ colorings that assign both vertices in the pair the same color[4]. So, the total number of colorings where two vertices in X are the same color is at most $\binom{|X|}{2} \cdot (k-1)^{n-1}$. Since $|X| \le \ell$, we can bound it by

$$\binom{|X|}{2} \cdot (k-1)^{n-1} \le \binom{\ell}{2} \cdot (k-1)^{n-1} \le \ell^2 \cdot (k-1)^{n-1}.$$

So, at most $\ell^2 \cdot (k-1)^{n-1}$ inputs from $\mathcal{N}$ are rejected. Since there are a total of $(k-1)^n$ graphs with colorings in $\mathcal{N}$, the fraction of accepted graphs is at least

$$1 - \frac{\ell^2 \cdot (k-1)^{n-1}}{(k-1)^n} = 1 - \frac{\ell^2}{k-1}.$$

Since the approximator is an OR of indicators, the approximator accepts at least as many No graphs. □

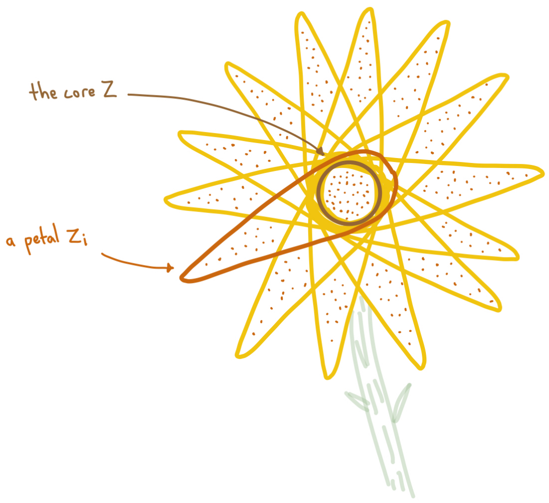
Sunflowers in our Toolbox

In our construction, we will see that we will need to somehow decrease the number of clique indicators in an approximator. We will see that the Sunflower Lemma will be an incredibly useful tool here. This is a purely combinatorial result discovered by Erdös and Rado in 1960.

[4]This is because we can consider the pair as one element, so we have $k-1$ options for $n-1$ elements to color.

Definition 8: A collection of sets $\{Z_1, \dots, Z_p\}$ is called a *sunflower* if there exists a set Z (which we call the *core*) such that for any $i \neq j$, we have $Z_i \cap Z_j = Z$. The sets Z_i are called the *petals* of the sunflower[5]. Equivalently, every element either belongs to none, one, or all of the sets in the collection. ◁

Note that a collection where all sets are disjoint from each other is also a sunflower with an empty core.

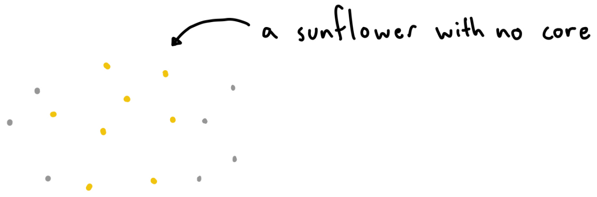


Lemma 2 (Sunflower): Let $\mathcal{T}$ be a collection of non-empty sets each of size at most ℓ. If $\mathcal{T}$ contains more than $\ell!(p-1)^\ell$ sets, then it contains a sunflower with p petals. ◁

Intuitively, this means that large collections of small sets have sunflowers.

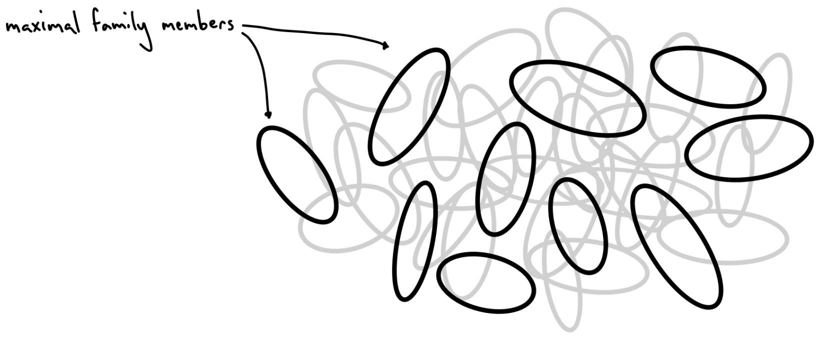
Proof (Lemma 2): We will prove this lemma by induction on ℓ.

For the base case, if $\ell = 1$, then all sets must contain only one element. If $|\mathcal{T}| > \ell!(p-1)^\ell = p-1$, then we can choose any p sets in $\mathcal{T}$. This is a sunflower with the empty set as its core.



[5]Here, we recklessly call the sets in the sunflower petals, even though real sunflower cores aren't considered to be part of their petals.

Now, let's show the inductive step. Let $\ell \geq 2$. Suppose that we know that the lemma holds for $\ell - 1$. Let's take a maximal family[6] of pairwise disjoint sets in $\mathcal{T}$. Let the sets in the family be called $X_1, \ldots, X_t$ and let $X = X_1 \cup \cdots \cup X_t$.



If $t \geq p$, then we are done because any p sets from our family form a sunflower with an empty core.

If $t < p$, then we can see that

$$|X| = \sum_{i=0}^{t} |X_i| \leq \sum_{i=0}^{t} \ell = \ell(p-1).$$

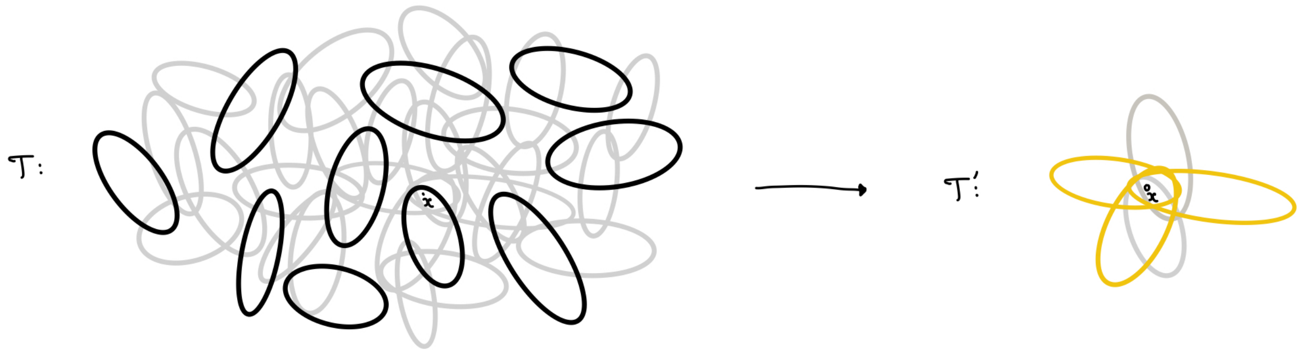
Since our family is maximal, every set in $\mathcal{T}$ must intersect with X (otherwise we would be able to add it to the collection). There are more than $\ell!(p-1)^\ell$ sets in $\mathcal{T}$ and only $\ell(p-1)$ points in X, by the pigeonhole principle, there exists some point $x \in X$ that is contained in at least

$$\frac{|\mathcal{T}|}{|X|} \geq \frac{\ell!(p-1)^\ell}{\ell(p-1)} = (\ell-1)!(p-1)^{\ell-1}$$

of the sets in $\mathcal{T}$. If we remove x from all of the sets containing it, then those sets form a collection on which we can use our inductive hypothesis. More formally, if we take the set

$$\mathcal{T}' = \{S \setminus \{x\} : S \in \mathcal{T}, x \in S\},$$

then all sets in $\mathcal{T}'$ have at most $\ell - 1$ elements, so by the inductive hypothesis, there is a sunflower with p petals in $\mathcal{T}'$.



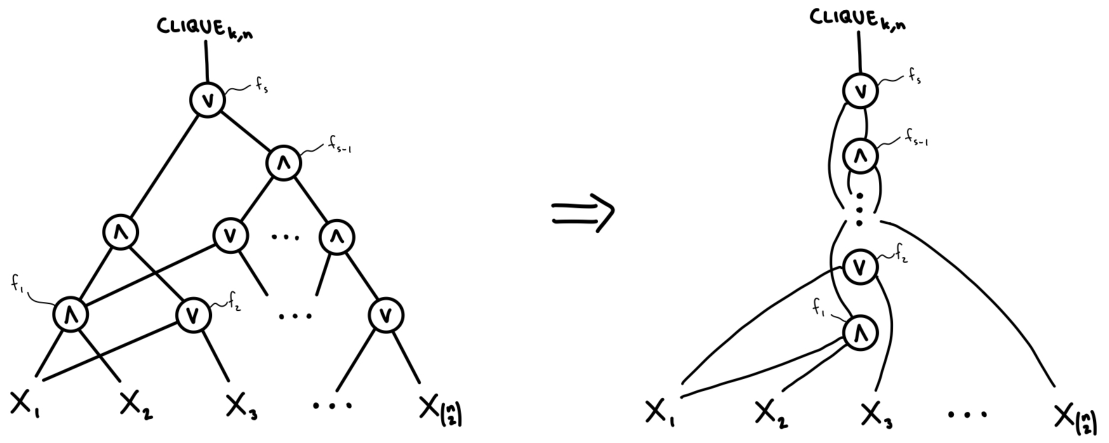
If we add x back to all of the sets in this sunflower, then we still have a sunflower with p petals, but now all sets are in $\mathcal{T}$. This is what we wanted to prove existed, so we are done. □

[6]Here, maximal means that it's impossible to add another set to our family while maintaining the property that all sets are pairwise disjoint.

Small Circuits Have Good Approximators

Now, let's dive into the actual construction of our approximator. As mentioned earlier, we assume that there is a small monotone circuit computing $\textsc{Clique}_{k,n}$ and construct an (m, ℓ)-approximator that approximates this circuit well on Yes and No graphs. For each gate in the circuit, we will build an (m, ℓ)-approximator for the function it computes.

Let F be a monotone circuit of size s that computes $\textsc{Clique}_{k,n}$. Let us order the gates of F in a way that parent gates always come after their children. Then we can consider the sequence of functions $f_1, f_2, \ldots f_s$ where f_i is the function computed by the i^{th} gate. So, $f_s = \textsc{Clique}_{k,n}$ and each f_i is either an OR or an AND of variables or functions that come before f_i in the sequence.



We can define the approximator for the input variable x_{ij} to simply be $C_{\{i,j\}}$. Then, for each i, if f_i is an OR of two functions (or input variable), then we let the approximator for f_i be $A \sqcup B$ where A and B are the approximators for the two child functions of f_i. If f_i is an AND of two functions (or input variable), then we let the approximator for f_i be $A \sqcap B$ where A and B are the approximators for the two child functions of f_i.

We haven't yet defined what the $\sqcup$ and $\sqcap$ are, so let's explore what they should be.

Taking the OR

Let's start with the $\sqcup$ operator. We have two (m, ℓ)-approximators,

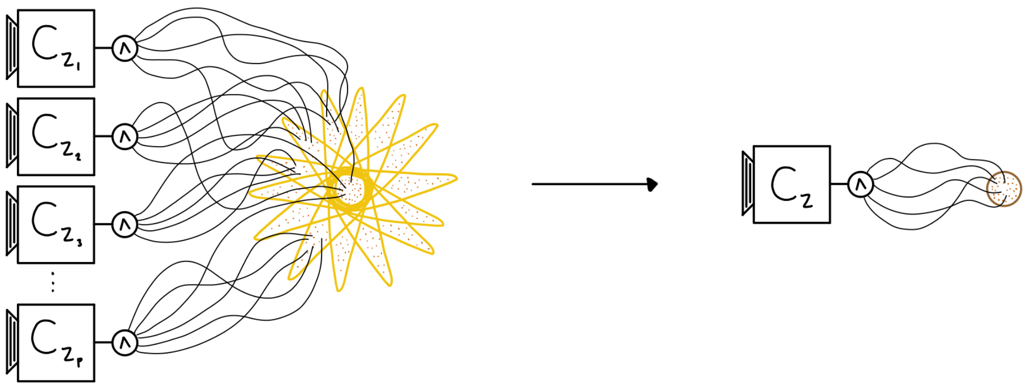
$$A = \bigvee_i C_{A_i} \text{ and } B = \bigvee_i C_{B_i},$$

and our goal is to create an (m, ℓ)-approximator for the OR of the two. It would be nice if our new approximator could simply be

$$\bigvee_i C_{A_i} \bigvee_i C_{B_i}.$$

This is an approximator, but the number of indicators it is made up of can be as high as $2m$. So, we somehow need to go from a $(2m, \ell)$ approximator back to an (m, ℓ)-approximator.

This is where the Sunflower Lemma comes to save us. We will show that if we have an OR of a bunch of indicators whose sets that form a sunflower, then we can replace all of these by just an indicator for the core without too big of an error.



So, as long as we have enough sets for the condition of the Sunflower Lemma, we can repeatedly replace sunflowers with their cores to reduce the number of indicators in our approximator. Namely, if we set $m := \ell!(p-1)^\ell$ for some p, then we can repeatedly decrease the number of indicators by $p-1$ until we have no more than m indicators in our approximator total. We call this process *plucking*. With this method, we have created an (m,ℓ)-indicator that approximates an OR of two other (m,ℓ)-indicators. We will prove that the approximation is good enough for our purposes later. Also, now that we have expressed m through this new parameter p, we will be carefully selecting p later instead of selecting m. We summarize this idea with the following definition.

Definition 9: Given two (m,ℓ)-approximators A and B, we define $A \sqcup B$ to be the (m,ℓ)-approximator created by taking the OR of all indicators in either A or B and then plucking sunflowers until the number of indicators is no more than m. ◁

Taking the AND

Now, let's think about the AND case. Again, we have two (m,ℓ)-approximators,

$$A = \bigvee_i C_{A_i} \text{ and } B = \bigvee_i C_{B_i},$$

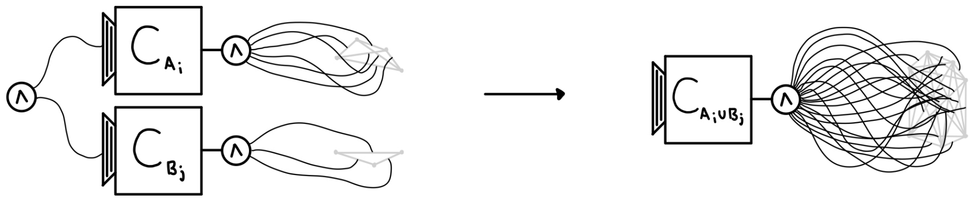
and our goal is to create an (m,ℓ)-approximator for the OR of the two. In an ideal world, we would want an (m,ℓ)-approximator that computes $A \wedge B$, which can be expanded to

$$A \wedge B = \left(\bigvee_i C_{A_i}\right) \wedge \left(\bigvee_i C_{B_i}\right) = \bigvee_{i \neq j} \left(C_{A_i} \wedge C_{B_j}\right).$$

Unfortunately, there are several reasons why this isn't an (m,ℓ)-approximator. First, this isn't actually an OR of clique indicators. To deal with this, we can approximate $C_{A_i} \wedge C_{B_j}$ with $C_{A_i \cup B_j}$. So, we can write

$$A \wedge B \approx \bigvee_{i \neq j} C_{A_i \cup B_j},$$

which is indeed an OR of clique indicators.



Now we encounter the problem that $A_i \cup B_j$ might be a set of size more than ℓ, so our indicators are too big. To handle this, we can simply throw out all indicators that are over a set that has more than ℓ elements. Like in the OR case, we may also end up with too many of these indicators in our OR (up to m^2, in fact), but we can use the exact same plucking method to reduce this number back to at most m. Our result is now guaranteed to be an (m, ℓ)-approximator. We can summarize the AND construction with the following definition.

Definition 10: Given two (m, ℓ)-approximators A and B, we define $A \sqcap B$ to be the (m, ℓ)-approximator created by taking the OR of clique indicators for all possible unions of an indicator set from A with an indicator set from B, then removing the indicators over sets with more than ℓ elements, and finally plucking sunflowers until the number of indicators is no more than m. $\triangleleft$

So, we now know how to build our way up the gates of F and construct an approximator for all f_i, including $f_s = \textsc{Clique}_{k,n}$. Our construction guarantees that the result is an (m, ℓ)-approximator, but we still need to prove that the approximation is good on inputs from $\mathcal{Y}$ and $\mathcal{N}$. To do this, we will show that the number of Yes and No graphs that are misinterpreted at each construction step is small, and then take the union bound to get the maximum error of the final approximator in terms of s, n, k, ℓ, and p. This combined with Lemma 1 will allow us to choose ℓ and p to give a lower bound for s.

Good at Saying Yes

First, we see how good our final approximator is at handling Yes graphs. Let's denote the final (m, ℓ) approximator for f_s by $\mathring{F}$.

Lemma 3: The number of $\text{Yes}_{k,n}$ graphs that $\mathring{F}$ wrongly rejects is at most

$$s \cdot m^2 \cdot \binom{n - \ell - 1}{k - \ell - 1}.$$

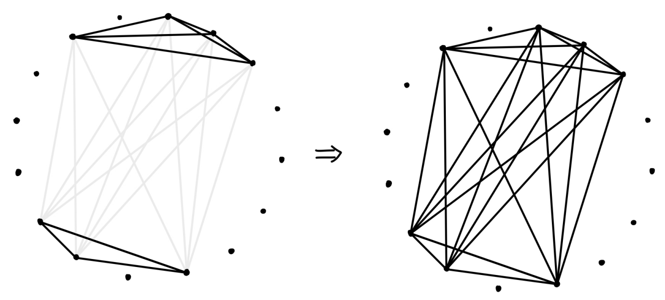
$\triangleleft$

Proof (Lemma 3): As mentioned earlier, we will show that at each step in the construction, the approximator f_i for some gate incorrectly outputs 0 on at most $m^2 \cdot \binom{n-\ell-1}{k-\ell-1}$ Yes graphs that were correct before this step. Then, we'll use the union bound to pick up the factor of s and prove the statement for the final result.

If f_i is approximating an OR gate, then any error could come only from the repeated plucking. However, we can see that when we pluck a sunflower, we replace the indicators over all its petals by an indicator over its core. Since the core is contained in all of the petals, a clique over a petal implies a clique over the core. So, plucking can only add to the cases where the approximator returns 1 and no errors are introduced in this case.

If f_i is approximating an AND gate, then there are three places in the construction that could cause an error.

The first place is when we are replacing each AND of indicators over two sets with an indicator over the union of the two sets. However, on Yes graphs, having a clique on two sets of vertices is the same as having a clique on the union of the two sets, no Yes graphs can be misidentified because of this.



The second place is when we are removing all indicators over sets of more than ℓ vertices. For a given indicator over a set of at least $\ell+1$ vertices, there are at most $\binom{n-\ell-1}{k-\ell-1}$ misidentified Yes graphs[7]. Since there are at most m^2 indicators in out construction that we could even throw out, the total number of misidentified Yes graphs on this step is no more than $m^2 \cdot \binom{n-\ell-1}{k-\ell-1}$, which is what we wanted to show.

The last possible place for error is when we are plucking the sunflowers, but this can only help accept a Yes graph by the same reasoning as before.

We can conclude by taking a union bound over the number of misidentified Yes graphs in each step to see that the total number of wrongly rejected Yes graphs is at most

$$s \cdot m^2 \cdot \binom{n-\ell-1}{k-\ell-1}.$$

$\square$

Good at Saying No

Similarly, we want to bound the error of our construction on No graph inputs. So, we prove the following lemma.

Lemma 4: The number of graphs (with colorings) from $\mathcal{N}_{k,n}$ that $\mathring{F}$ wrongly accepts is at most

$$s \cdot m^2 \cdot \ell^{2p} \cdot (k-1)^{n-p}.$$

$\triangleleft$

Note that we could express m in terms of p here to reduce the number of variables, but we'll avoid doing that to keep our formulas a little cleaner.

Proof (Lemma 4): Similarly to how we proved Lemma 3, we will bound the number of misidentified No graphs at each step by $m^2 \cdot \ell^{2p} \cdot (k-1)^{n-p}$ and take the union bound to show the result about the final approximator. Let's look at the approximator f_i at some step.

If f_i is approximating an OR gate, let us look at a random No instance G (color the vertices randomly and look at the resulting No graph). We want to show that each plucking only causes f_i to wrongly accept a small number of additional No graphs. Let's look at a specific plucking. Let $Z_1, \ldots, Z_p$ be the petals of the sunflower in this plucking and let Z be the core. For this plucking to cause an error, C_Z must wrongly accept G, but all C_{Z_i} should reject G. So, all vertices in Z should be colored differently but there should be two vertices colored the same in every Z_i.

Let's examine the probability of this happening. Let's call a set where all vertices are colored differently a rainbow set. Then,

$$\begin{aligned}
\Pr\left[Z \text{ is rainbow and all } Z_i \text{ are not rainbow}\right] &\le \Pr\left[\text{All } Z_i \text{ are not rainbow} \mid Z \text{ is rainbow}\right] && (1)\\
&= \prod_{i=1}^{p} \Pr\left[Z_i \text{ is not rainbow} \mid Z \text{ is rainbow}\right] && (2)\\
&\le \prod_{i=1}^{p} \Pr\left[Z_i \text{ is not rainbow}\right] && (3)
\end{aligned}$$

Here, equation (2) comes from the fact that Z_i are disjoint outside of Z, so the events ere independent. Inequality (3) comes from the fact that a set is more likely to be rainbow if you're given that part of it is rainbow. We also know that the number of non-rainbow colorings of Z_i is at most

$$\binom{|Z_i|}{2} \cdot (k-1)^{n-1} \le \binom{\ell}{2} \cdot (k-1)^{n-1} \le \ell^2 \cdot (k-1)^{n-1}$$

[7]This is because the clique of a misidentified Yes graph must contain the $\ell+1$ or more vertices, and there are $k-\ell-1$ more vertices to select out of the remaining $n-\ell-1$.

by the same reasoning as in Lemma 1. There are $(k-1)^n$ colorings, so

$$\Pr\left[Z_i \text{ is not rainbow}\right] \leq \frac{\ell^2 \cdot (k-1)^{n-1}}{(k-1)^n} = \frac{\ell^2}{(k-1)}.$$

Combining this with inequality (3), we have

$$\Pr\left[Z \text{ is rainbow and all } Z_i \text{ are not rainbow}\right] \leq \left(\frac{\ell^2}{(k-1)}\right)^p = \ell^{2p} \cdot (k-1)^{-p}.$$

This means that the total number of colorings wrongly accepted because of this plucking is bounded by

$$\ell^{2p} \cdot (k-1)^{-p} \cdot (k-1)^n = \ell^{2p} \cdot (k-1)^{n-p}.$$

Since we remove at least one extra set with each plucking, there are at most m pluckings, so taking the union bound gives us that the number of No graphs wrongly accepted because of this step is at most $m \cdot \ell^{2p} \cdot (k-1)^{n-p}$.

Alternatively, if f_i is approximating an AND gate, then there are the same three places where we could have error as in Lemma 3.

First, when we are replacing each AND of indicators over two sets with an indicator over the union of the two sets, no errors can be introduced. This is because if the union of two sets is rainbow, then the two sets are both rainbow, so a rejection by both set indicators implies a rejection by the union indicator.

Second, removing clique indicators over sets with more than ℓ elements can only remove from the accepted graphs, so this cannot cause any errors on No graphs.

Finally, plucking the sunflowers can cause errors here. We know each plucking can cause at most $\ell^{2p} \cdot (k-1)^{n-p}$ errors, and there are at most m^2 pluckings, so the number of errors in this step is at most $m^2 \cdot \ell^{2p} \cdot (k-1)^{n-p}$.

In either case, the number of wrongly accepted No inputs because of f_i is no more than $m^2 \cdot \ell^{2p} \cdot (k-1)^{n-p}$. So, we can take the union bound over all i to conclude that the total number of No inputs $\mathring{F}$ wrongly accepts is at most $s \cdot m^2 \cdot \ell^{2p} \cdot (k-1)^{n-p}$. □

Conclusion

Now, we are prepared to arrive at the main result. To review, we have shown that for any monotone circuit computing $\textsc{Clique}_{k,n}$, we can construct an (m, ℓ)-approximator $\mathring{F}$ that does well on Yes and No graphs relative to the size of the circuit (Lemmas 3 and 4). We have also shown that (m, ℓ)-approximators, including $\mathring{F}$, either reject all graphs, or are bad at identifying No graphs (Lemma 1). Rejecting all graphs would mean the approximator is bad at identifying Yes graphs. To be able to meet this error, the size of the circuit must be large.

Let's formalize this statement and prove our goal.
Proof (Theorem 1): Let F be a monotone circuit computing $\textsc{Clique}_{k,n}$ with size s. Let $\ell = \lfloor \sqrt{k-1}/2 \rfloor$, let $p = \lfloor \sqrt{k} \log_2 n \rfloor$, and let $m = \ell!(p-1)^\ell$.

We can build a good (m, ℓ)-approximator $\mathring{F}$ for the circuit by the process we developed. By Lemma 1, $\mathring{F}$ either rejects all graphs or wrongly accepts at least $1 - \ell^2/(k-1) \geq 1/2$ of graphs (with colorings) from $\mathcal{N}$. Let's consider these two cases separately.

Case 1 ($\mathring{F}$ rejects all graphs): Here, we can use Lemma 3 to bound s. There are $\binom{n}{k}$ $\text{Yes}_{k,n}$ graphs and all of them must be rejected. However, by Lemma 3, at most $s \cdot m^2 \cdot \binom{n-\ell-1}{k-\ell-1}$ $\text{Yes}_{k,n}$ graphs are rejected. So,

$$s \cdot m^2 \cdot \binom{n-\ell-1}{k-\ell-1} \geq \binom{n}{k}.$$

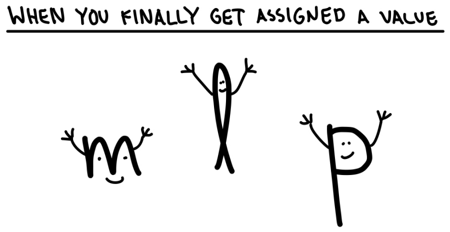
With some rearrangement and computations, it's possible to get from this that $s = n^{\Omega(\sqrt{k})}$. The computation is simple, but tedious and space-consuming, so we defer it until after the proof.

Case 2 ($\mathring{F}$ is wrong on half of No graphs): In this case, we have a similar idea. Half of $\text{No}_{k,n}$ graphs are wrongly accepted, but by Lemma 4, at most $s \cdot m^2 \cdot \ell^{2p} \cdot (k-1)^{n-p}$ of the No graphs are accepted. There are a total of $(k-1)^n$ $\text{No}_{k,n}$ graphs, so,

$$s \cdot m^2 \cdot \ell^{2p} \cdot (k-1)^{n-p} \geq \frac{1}{2} \cdot (k-1)^n.$$

From this, it is also possible to get that $s = n^{\Omega(\sqrt{k})}$.

Thus, in either case, $s = n^{\Omega(\sqrt{k})}$, so we are done. □



Bounds for the previous proof: Let us show the calculations that we claim are true in the preceding proof. Since both cases use m^2, let us bound that first.

$$\begin{aligned} m^2 &= (\ell!(p-1)^\ell)^2 \\ &\leq (\ell^\ell p^\ell)^2 = (\ell p)^{2\ell} = (\lfloor\sqrt{k-1}/2\rfloor\lfloor\sqrt{k}\log_2 n\rfloor)^{2\lfloor\sqrt{k-1}/2\rfloor} \\ &\leq (\sqrt{k-1}/2 \cdot \sqrt{k}\log_2 n)^{\sqrt{k-1}} \\ &\leq (\sqrt{k} \cdot \sqrt{k}\log_2 n)^{\sqrt{k}} = (k\log_2 n)^{\sqrt{k}} \\ &\leq (n^{1/4}\log_2 n)^{\sqrt{k}} \\ &\leq (n^{1/3})^{\sqrt{k}} = n^{1/3\cdot\sqrt{k}}, \end{aligned}$$

where the last inequality is only true for n greater than some constant[8].

A useful tool: We will use that

$$\begin{aligned} \frac{\binom{n}{k}}{\binom{n-t}{k-t}} &= \frac{n!(k-t)!(n-k)!}{k!(n-k)!(n-t)!} \\ &= \frac{n!(k-t)!}{k!(n-t)!} \\ &= \prod_{i=0}^{t-1} \frac{n-i}{k-i} \\ &\geq \prod_{i=0}^{t-1} \frac{n}{k} = \left(\frac{n}{k}\right)^t. \end{aligned}$$

Case 1: We have that

$$s \cdot m^2 \cdot \binom{n-\ell-1}{k-\ell-1} \geq \binom{n}{k}.$$

[8]Although this constant is quite large—somewhere around 2^{80}.

Rearranging, we have

$$\begin{aligned}
s &\geq \frac{1}{m^2} \cdot \frac{\binom{n}{k}}{\binom{n-\ell-1}{k-\ell-1}} \\
&\geq \frac{1}{m^2} \cdot \left(\frac{n}{k}\right)^{\ell+1} \\
&\geq \frac{1}{n^{1/3 \cdot \sqrt{k}}} \cdot \left(\frac{n}{k}\right)^{\ell+1} \\
&\geq \frac{1}{n^{1/3 \cdot \sqrt{k}}} \cdot \left(\frac{n}{n^{1/4}}\right)^{\ell+1} = \frac{1}{n^{1/3 \cdot \sqrt{k}}} \cdot \left(n^{3/4}\right)^{\ell+1} = \frac{1}{n^{1/3 \cdot \sqrt{k}}} \cdot \left(n^{3/4}\right)^{\lfloor \sqrt{k-1}/2 \rfloor + 1} \\
&\geq \frac{1}{n^{1/3 \cdot \sqrt{k}}} \cdot \left(n^{3/4}\right)^{\sqrt{k-1}/2} = n^{3/8 \cdot \sqrt{k-1} - 1/3 \cdot \sqrt{k}} \\
&\geq n^{1/1000 \cdot \sqrt{k}},
\end{aligned}$$

where the last inequality holds for k larger than some constant[9]. So, $s = n^{\Omega(\sqrt{k})}$ in this case.

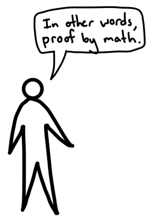
Case 2: We have that

$$s \cdot m^2 \cdot \ell^{2p} \cdot (k-1)^{n-p} \geq \frac{1}{2} \cdot (k-1)^n.$$

Rearranging, we have

$$\begin{aligned}
s &\geq \frac{1}{m^2} \cdot \frac{1/2 \cdot (k-1)^p}{\ell^{2p}} \\
&= \frac{1}{m^2} \cdot \frac{1/2 \cdot (k-1)^{\lfloor \sqrt{k} \log_2 n \rfloor}}{\lfloor \sqrt{k-1}/2 \rfloor^{2 \lfloor \sqrt{k} \log_2 n \rfloor}} \\
&\geq \frac{1}{m^2} \cdot \frac{1/2 \cdot (k-1)^{\lfloor \sqrt{k} \log_2 n \rfloor}}{(\sqrt{k-1}/2)^{2 \sqrt{k} \log_2 n}} \\
&= \frac{1}{m^2} \cdot \frac{1/2 \cdot (k-1)^{\lfloor \sqrt{k} \log_2 n \rfloor}}{((k-1)/4)^{\sqrt{k} \log_2 n}} \\
&= \frac{1}{m^2} \cdot \frac{1}{2} \cdot (k-1)^{\lfloor \sqrt{k} \log_2 n \rfloor - \sqrt{k} \log_2 n} \cdot 4^{\sqrt{k} \log_2 n} \\
&\geq \frac{1}{m^2} \cdot \frac{1}{2} \cdot (k-1)^{-1} \cdot 4^{\sqrt{k} \log_2 n} \\
&= \frac{1}{m^2} \cdot \frac{1}{2} \cdot (k-1)^{-1} \cdot n^{2\sqrt{k}} \\
&\geq n^{-1/3\sqrt{k}} \cdot \frac{1}{2} \cdot n^{-1/4} \cdot n^{2\sqrt{k}} \\
&= \frac{1}{2} \cdot n^{(2-1/3)\sqrt{k} - 1/4} = n^{\Omega(\sqrt{k})}.
\end{aligned}$$

So, $s = n^{\Omega(\sqrt{k})}$ in both cases and we are done!



[9] About 5. With some manipulation, it's possible to decrease these constants, but we will not do that here.

Sources

I owe my ability to write these notes to the following sources.

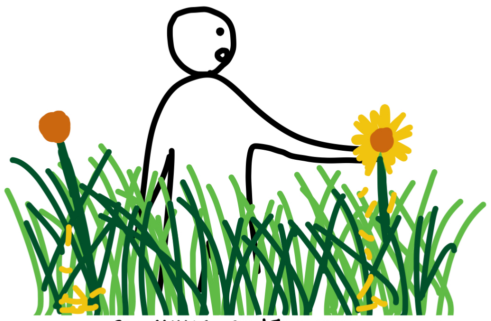
Boaz Barak and Sanjeev Arora's *Computational Complexity: A Modern Approach*, 2007,

Stasys Jukna's *Boolean Function Complexity: Advances and Frontiers*,

Samuel R. Buss's *Scribe notes on Razborov's bound*, 2013,

Éva. Tardos' *The Gap Between Monotone and Non-monotone Circuit Complexity is Exponential*,

Li-Yang Tan and his amazing complexity theory courses.



PLUCKING SUNFLOWERS