PAUL BEAME

CSE 599S COURSE NOTES: PROOF COMPLEXITY AND ITS APPLICATIONS

Contents

1
Introduction to Proof Complexity

1.1 Origins and motivation

The study of proofs was essential to the birth of computer science, and the desire to automate mathematics in a bug-free way led to its formal models and early development. Since then, a focus on obtaining practical algorithms for finding proofs has continued to be an important thread in the development of the field. This has particularly risen to prominence in the substantial practical advances in formal methods in the last two decades.

Proofs have played another important role in computer science in the theory of NP and NP-completeness. In particular, Cook's 1971 paper introducing these notions is entitled "The Complexity of Theorem-Proving Procedures" and was inspired in part by the work of Davis and Putnam on the search for proofs, . Indeed, proofs are involved in the standard intuitive characterization of NP: NP consists of all languages L that have "short, easy-to-check, proofs of membership"; that is, we can verify in polynomial time in $|x|$ that $x \in L$, when given a short proof string y (where short means polynomial in $|x|$).

Example 1.1. What does it mean to give a "proof" that a number such as 1906001 is composite? The number itself contains all the information required and we could simply run a factoring algorithm to check this fact. On the other hand, writing 1906001=1009x1889 really is a proof that we can easily check. The key property that makes the latter a proof is that *we can check it in polynomial time*.

How does the NP notion of proof compare with the notions of proof that we are familiar with from logic and discrete mathematics? There, rather than the satisfiability problem that characterizes NP, one considers problems of tautology or unsatisfiability, both coNP-complete problems. In logic, proofs are objects like truth tables, or sequences of inferences according to some formal system. More generally,

- ▹ *What do we mean to say that something is a proof even when, like a truth table, it is much longer than the thing being proved?*

In their foundational 1979 paper introducing the notions of proof complexity, Cook and Reckhow pointed out that the right notion is that a string of symbols is a proof precisely if it is possible to check it in time polynomial in its length. A basic question for proof complexity roughly is the following:

- *Given a true statement A, how long a proof do we need to prove A?*

As stated, this question doesn't quite make sense since proof strings themselves have no independent meaning without understanding how they will be checked. Cook and Reckhow noted that the question needs to be studied separately for each correct format for writing down proofs or *proof system*. Therefore, the proper version of the basic question of proof complexity is:

- *Given a true statement A and proof system S, how long is the shortest proof in S that A is true?*

Building on this basic question, some natural questions that Cook and Reckhow raised were

- *How do different proof systems compare with respect to proof length?*
- *Is there a proof system S that is* efficient *in that for any true statement A, the length of the shortest proof in S that A is true is polynomial in the length of A?*

It is not hard to see that an efficient proof system would be an NP algorithm. In particular, since for the set of propositional logic tautologies $TAUT$ is coNP-complete, efficient proof systems for $TAUT$ exist if and only if $\mathsf{NP} = \mathsf{coNP}$.

One can see, for example, that truth tables form a very inefficient proof system. In this course we will study many other natural proof systems for propositional logic, some because they have been important in practice, and others because they have nice theoretical properties. Most of these systems are organized around a framework of axioms and inference rules of the sort that one finds in logic texts.

A key part of proof complexity is in producing lower bounds on the lengths of proofs in various proof system. Here, one can view finding superpolynomial proof length lower bounds for increasingly powerful proof systems as stepping stones towards proving that $\mathsf{NP} \neq \mathsf{coNP}$.

Though many proof complexity questions focus on *nondeterministic* algorithms, proof complexity also analyzes methods for finding short proofs if they exist. Since, for example, the statement A could be "This code is bug-free" or "No solution to these constraints has value better than K", proof complexity has applications to many areas of computer science, including optimization and formal verification among others.

Proof complexity methods also give us tools to analyze the power and limitations of entire classes of algorithms. For example:

- Modern SAT solvers that are at the heart of much of current work in formal methods in PL/SE can be understood in terms of resolution proofs.
- Semi-definite programming based on sum-of-squares proofs is the best algorithm we know for solving large classes of optimization problems. In this case, the mere existence of a short proof that there is a solution is already enough to guarantee that there is a good algorithm to find that solution.

From a practical point of view, large lower bounds on the sizes of proofs yield lower bounds on the time to find them – just writing them down takes a large amount of time. In particular, lower bounds for finding specific kinds of proofs in proof systems that underlie practical solution methods have influenced the directions for the use of these methods in practice, and proof complexity methods are now part of practical developments in these areas.

1.2 Proof complexity definitions

We give a definition of proof systems that is closely based on the definition of NP.

Definition 1.2. A *proof system* for a language L is a polynomial-time (verifier) algorithm V such that

$$x \in L \Leftrightarrow \exists y.\, V(x, y).$$

The string y is a *proof* in proof system V that $x \in L$.

The only difference with NP is that there is no bound on $|y|$ as a function of $|x|$. If $L = TAUT$, the statement $x \in L \Leftrightarrow \exists y.\, V(x, y)$ is equivalent to saying that proof system V is *sound* ($\exists y.\, V(x, y) \Rightarrow x \in L$) and *complete* ($x \in L \Rightarrow \exists y.\, V(x, y)$) in the usual terminology of proof methods.

Definition 1.3. The *complexity* of a proof system V is the smallest function $S : \mathbb{N} \to \mathbb{N}$ such that

$$x \in L \Leftrightarrow \exists y.\, |y| \leq S(|x|).\, V(x, y).$$

Proof system V is *polynomially bounded* iff this complexity bound $S(n)$ is $n^{O(1)}$.

The following is immediate from the definition.

Proposition 1.4. *$L \in$ NP if and only if L has a polynomially-bounded proof system.*

More generally, the strength of a proof system is measured by how short its proofs can be – stronger proof systems are associated with shorter proofs.

Definition 1.5. A *propositional proof system* is a proof system V for $TAUT$; that is,

$$F \in TAUT \Leftrightarrow \exists P.\ V(F,P)$$

Alternatively, we can define propositional proof systems in terms of refutations; that is, a propositional proof systems a proof system for $UNSAT$, the set of unsatisfiable Boolean formulas.

Since both $UNSAT$ and $TAUT$ are coNP-complete, we immediately obtain the following:

Theorem 1.6. *$NP = coNP$ if and only if there is a polynomially bounded propositional proof system.*

Remark 1.7. In their original paper, Cook and Reckhow defined a proof system for a non-empty language L to be a polynomial-time computable function f from the set of all strings *onto* the set L; the pre-images of a string x under f are the proofs y that $x \in L$.

This definition is essentially equivalent to the one from Definition 1.2, though the two formally look quite different. The only major difference is that each proof y in the Cook-Reckhow definition must explicitly encode the x that it proves is in L.

For one direction, given a proof system V as above, define $f_V(x,y) = x$ if $V(x,y)$, and $= x_0$ for some fixed $x_0 \in L$ otherwise. For the other direction, given a Cook-Reckhow proof system f, define the verifier V_f on input (x,y) to simply check that $x = f(y)$.

In general, while there are aesthetic reasons to prefer one or the other definition, there are no aesthetic differences in the case of the proof systems that we will consider.

Definition 1.8. We say that proof system U *polynomially simulates* proof system V iff

- U and V prove the same language L, and
- proofs in V can be efficiently converted into proofs in U; that is, there is an polynomial-time algorithm that given (x,y) such that $V(x,y)$ produces a y' such that $U(x,y')$.

Proof systems U and V are *polynomially equivalent* iff they polynomially simulate each other.

The partial order given by polynomial simulation is a natural measure of the relative strength of different proof systems.

In our focus on propositional proof systems, it will be convenient to focus on proof systems for unsatisfiability of CNF formulas which, by the standard reduction of $CIRCUIT\text{-}SAT$ to $CNFSAT$, immediately yields corresponding proof systems for $UNSAT$ and $TAUT$.

In particular, given a propositional logic formula F, define a CNF formula F' where we

- add an extra variable y_G for each distinct subformula G of F
- include clauses to F' expressing that y_G takes on the value of G determined by the inputs of F, in particular
  - if $F = G \vee H$ then include $(\neg y_F \vee y_G \vee y_H), (\neg y_G \vee y_F), (\neg y_H \vee y_F)$,
  - if $F = G \wedge H$ then include $(\neg y_F \vee y_G), (\neg y_G \vee y_H), (\neg y_G \vee \neg y_H \vee y_F)$,
  - if $F = \neg G$ then include $(\neg y_F \vee \neg y_G), (y_F \vee y_G)$
- include clause y_F to express that F must evaluate to true.

The following proposition then follows immediately from this construction.

Proposition 1.9. $F \in UNSAT \iff F' \in CNF\text{-}UNSAT$.

For the remainder we will assume, unless otherwise noted, that the propositional proof systems we consider are proof systems for $CNF\text{-}UNSAT$. Hence, the inputs for our proof systems will be CNF formulas and our proofs will be refutations.

There are many different propositional proof systems – some that we don't even normally think of in terms of proofs. In particular *any* correct algorithm A that *decides* satisfiability of CNF formulas also yields a propositional proof system. A proof for a formula F in this case is a correct transcript of the execution of A on input F. A verifier V_A can easily check whether a string P is a transcript of A on input F and that this transcript says that A outputs that F is unsatisfiable.

Why propositional *proof complexity?* There are several reasons to focus on propositional proof systems:

- Propositional proofs lie at the heart of approaches to algorithmic reasoning and formal methods for more complex questions. In particular, reduction of these complex questions to SAT solving underlies many of the most successful practical formal methods today. As an example of the potential of this approach, Davis and Putnam's 1960 paper already showed how to convert theorem proving for first-order logic to a series of propositional proof complexity questions on the unsatisfiability of CNF formulas.
- Lower bounds on the complexity for specific proof systems of increasing strength can be seen as stepping stones towards proving $\mathsf{NP} \neq \mathsf{coNP}$ and hence $\mathsf{P} \neq \mathsf{NP}$.
- Understanding specific proof systems allows us to understand broad classes of algorithms for NP-hard problems already in use in practice and to suggest new approaches to solving them.

A dynamic framework for proof systems Before going through specific proof systems, we discuss a general framework shared by many propositional proof systems: In these proof systems, proofs are expressed as sequences of objects $F_1, F_2, \ldots, F_t$ called proof *lines*. These lines are often Boolean formulas but can be other kinds of objects. In these proof systems, there are rules for introducing lines independent of the previously lines, called *axioms*, and rules for introducing lines based on some bounded number of previous lines, called *inference rules*. We call these proof systems *dynamic* because we can view each proof as an increasing sequence of lines.

For example, in proof systems for tautologies whose lines are Boolean formulas, the axioms might be based on axioms such as

$$\text{(Excluded Middle)} \qquad \frac{}{A \vee \neg A}$$

which lets one introduce any line of the form $A \vee \neg A$, and inference rules such as

$$\text{(Modus Ponens)} \qquad \frac{A \qquad A \to B}{B}$$

which says that from lines A and $A \to B$ one can introduce/infer/derive) the line B. Each of these axioms or inference rules is a *schema* in that A and B refer to arbitrary formulas. A proof of F is correct if the last formula $F_t = F$ and the axioms and inference rules are correctly applied.

In the case of proofs of unsatisfiability – *refutations* – one starts with axioms derived directly from the input formula F and the final line F_t is some fixed marker of obvious unsatisfiability associated with the proof system.

Proof verification for both of these kinds of systems involves checking that the various axioms and inference rules are followed appropriately and bear the right relationship to the input formula F.

Every dynamic proof P implicitly defines a directed graph of inferences of P whose vertices are the lines P, and whose directed edges go from each proof line to the previous lines involved in the inference rule deriving it. (Alternatively, we could direct all edges in the opposite direction.) This directed graph is acyclic with a topological sort given by the (reverse of the) sequence of lines, hence we call it a *proof DAG*.

For every dynamic proof system S, it turns out to be natural to consider the restricted version in which the proof DAG is a tree, which we call *tree-like S*, or simply *tree S*. Since these dynamic proof systems only require local consistency properties, it is always possible to expand a dynamic proof to a tree-like one by repeating lines. Though they are less efficient in general, one reason to consider tree-like proofs is that they can be much easier to search for, and another is that the study of such proofs gives a stepping stone towards understanding the general case.

1.3 Logic-based propositional proof systems

Truth tables

Truth tables are easy to check (in fact, checkable in linear time) but since an n variable formula F requires a truth table of size at least $|F| \cdot 2^n$, they are extremely inefficient[1].

[1] It seems hard to believe that one can do any worse but there are natural well-studied proof systems that are even less efficient than truth tables. For example, the propositional part of *analytic tableaux*, an intuitive proof system that is the most popular proof system for understanding more general modal logics, requires $n!$ size proofs for small n variable ordinary propositional formulas in the worst case.

Resolution

Resolution is a dynamic refutation system for CNF formulas. Resolution and its special cases form the most widely used and studied family of proof systems, both in practice and theoretically.

- The lines of a resolution proof are clauses, which are viewed as unordered sets of literals.
- The axioms of a resolution refutation of CNF formula F are the clauses of F.
- Resolution has a single inference rule

$$\text{(Resolution)} \qquad \frac{A \vee x \qquad B \vee \neg x}{A \vee B}$$

  where A and B are sets of literals and x is a variable. We say that $A \vee B$ is the *resolvent* of $A \vee x$ and $B \vee \neg x$ and that these two clauses are *resolved on* x to produce $A \vee B$. Soundness follows because a truth assignment to x can satisfy at most one of $A \vee x$ or $B \vee \neg x$.
- A resolution derivation is a refutation iff it derives the empty clause, which we denote by $\bot$.

Like all dynamic proof systems, it is natural to consider the subclass of resolution proofs whose inference graph is a tree.

There are two other subclasses of resolution proofs that are important.

Definition 1.10. A resolution proof is *regular* iff on any path in the proof DAG each literal occurs at most once.

Lemma 1.11. *Any minimal length tree resolution proof is regular.*

Proof. The main idea is that if there are two clauses on a path that resolve on variable x, then one can remove the resolution step that is further from the root and simplify the remainder of the path without increasing the proof size. The details are left as an exercise. □

From now on, we assume without loss of generality that tree resolution proofs are regular.

Regular resolution is the proof system in which only regular proofs are allowed. This seems a natural restriction: Since the goal is to end up with the empty clause, it seems intuitively wasteful to remove literals only to add them back again later. (However, the analogue of Lemma 1.11 does not hold in general.)

Davis and Putnam's 1960 paper considered a class of resolution proofs in which variables are eliminated one by one in some order. Each variable is eliminated by replacing all clauses involving that variable by the set of all possible resolvents using those clauses. This class of proofs corresponds to the subclass of regular resolution proofs called *ordered resolution* in which the order of appearance of variables resolved on every path in the proof DAG is consistent with some fixed total order σ.

It turns out that tree resolution and ordered resolution are incomparable proof systems.

Tree resolution and DPLL

We will see that tree resolution corresponds to a natural backtracking algorithm to search for satisfying assignments that was introduced in a 1962 paper by Davis, Logemann, and Loveland, which built on the 1960 paper by Davis and Putnam mentioned earlier. After some confusion in the literature about how to refer to this algorithm (it was sometimes erroneously called the Davis-Putnam algorithm, though their 1960 paper had the different resolution-based variable elimination algorithm mentioned above), the terminology has settled on the acronym *DPLL*.

The algorithm uses the notion of the simplification of a CNF formula F by assigning a Boolean value to a literal. Given a literal x we define $F_{x\leftarrow 1}$ to be the simplification of F in which every clause containing x is removed (because it is satisfied) and every clause containing $\neg x$ has $\neg x$ removed; $F_{x\leftarrow 0}$ is the same as $F_{\neg x\leftarrow 1}$. By construction, F and $F_{x\leftarrow 1}$ agree on all assignments on which x has value 1.

Algorithm 1: The DPLL algorithm for satisfiability search. This is invoked as DPLL(F,nil). This is a *complete* algorithm in that failure of the search implies that F is unsatisfiable.

```
1:  function DPLL(F, A)
2:      while F contains a clause x of size 1 do
3:          F ← F_{x←1}; A ← (A, x)                ▷ Unit propagation
4:      if F is empty then
5:          Halt and output satisfying assignment A
6:      if F contains the empty clause ⊥ then
7:          return
8:      else choose unset literal x                 ▷ Decision literal
9:          DPLL( F_{x←1} , (A, x))
10:         DPLL( F_{x←0} , (A, ¬x))
```

Note that DPLL as stated is really a family of algorithms since the choice

of unset literal x in step 8, the *branching step*, requires a choice, which is governed by a *literal selection heuristic*. At each step of DPLL, the simplified version of F is called the *residual formula*. At each line, the residual formula is equivalent to the original formula on every assignment consistent with the literals in A, which is called the *trail* of the search. Each literal x in A is associated with a *decision level* which is the number of decision literals in the prefix of A up to and including x.

Observe that the *trace* of the execution of a failed DPLL search for a satisfying assignment on input F is a proof that F is unsatisfiable. (The associated proof checker can simply check that the trace follows the steps of the algorithm.) The running time of DPLL is at least the number of times that the pair (F,A) are re-assigned in the algorithm.

Observe that if we remove unit propagation lines 2-3 and instead in line 8 choose a literal in a clause of size 1, if one exists, in line 9. then the call on line 11, if reached, would immediately return because $F_{x\leftarrow 0}$ would contain an empty clause. This would be essentially the same length execution, though of course it would be a constant factor slower. We call the version of DPLL in which lines 2-3 are removed and the literal selection heuristic is arbitrary, *nondeterministic DPLL*. When we discuss DPLL as a proof system, we will assume that it is this nondeterministic version, whose shortest computation is a lower bound on all deterministic DPLL algorithm runtimes.

Theorem 1.12. *DPLL and tree resolution are equivalent proof systems.*

Proof. More precisely, we show that DPLL executions on F that fail to find a satisfying assignment correspond to (regular) tree resolution refutations of F in which branching steps of the DPLL execution involving variable x correspond to with tree resolution steps resolving on variable x, and vice versa.

First, given a regular tree resolution refutation of F, we replace each resolution on variable x, by a branch on literal x (the order in which the two child subtrees are followed is irrelevant since both are visited in the DPLL search). The subtree where $x \leftarrow 1$ is associated with the clause that contains $\neg x$ and that with $x \leftarrow 0$ is associated with the clause containing x. This maintains the fact that the partial assignment at a node n the DPLL tree falsifies the corresponding clause in the tree resolution proof. In particular, since all the variables from an input clause of the tree resolution proof are resolved away, the partial assignment associated with a leaf falsifies the associated input clause and hence the DPLL computation will have an empty clause at each leaf as required.

Conversely, consider the recursion tree of the (nondeterministic) DPLL execution. At each leaf of this tree, there is a clause of the residual formula that is empty. This leaf corresponds to an original clause of F that has been made false by the assignment A at the node. Make a copy of the this tree, labeling each leaf by the original clause that produced it. In filling out the rest

of the tree resolution refutation, we maintain the invariant that the assignment A associated with a node of the DPLL tree falsifies the corresponding node of the tree resolution proof. In particular, the node corresponding to a branching node on variable x by the resolvent of the two child clauses on x if both child clauses contain x and, otherwise, we simply copy one of the child clauses without x. □

DPLL implementation: Watched literals As stated, the explicit computation of each residual formula can be quite time-consuming. In practice, the residual formula is not computed explicitly. Observe that it is only necessary to know which residual clauses have size 1 (or become empty). To do this a *watched literal* technique is used: For each clause, the data structure keeps two pointers to distinct literals, the watched literals, initially the first two literals in the clause, along with a list of clauses for which these are watched; in addition, a flag is kept for each clause to indicate whether or not it is active.

When an assignment of literal x is made, only clauses for which that variable is watched are examined; all other clauses will have residual size at least two and do not need to be considered. All clauses for which x is the watched literal are marked inactive. For each active clause for which $\neg x$ is a watched literal, the next literals after the last watched literal are examined, one after another, until an unset literal is found. If one is found, it replaces $\neg x$ as the watched literal. Otherwise, the clause has been reduced to a unit clause and the other watched literal is propagated.

The watched literal method ensures that propagation can be done efficiently. It becomes even more important for the extension of DPLL algorithms, known as *Conflict-Directed Clause Learning (CDCL) SAT solvers*, are implemented, since these solvers add many clauses, some of them potentially long, to the original formula F during computation.

CDCL SAT solvers

These are the most important practical algorithms for SAT solving and formal reasoning. In DPLL, when the search fails because of a conflict in line 7, the recursive calls simply backtrack and the last decision is simply undone. However, while the conflict was found using the last branching decision, it may not depend on any other recent branching decision, so changing those decisions may not impact the conflict. The general idea of conflict-directed clause learning, is to record somewhat more about the decisions and unit propagations make during the proof search, using a data structure called a *conflict graph* and replace line 7 of DPLL with a *conflict analysis* step which adds a new *learned clause* to F, which summarizes the reason for the conflict and can be used to simplify future searches.

The *conflict graph* is a directed graph with one vertex for each literal

in the trail A. Decision literals are source nodes. For every literal x in A assigned through propagation of a unit clause that was originally clause C of F, all of the other literals z in C must have previously appeared as $\neg z$ in A; in the conflict graph we put an edge from each of these $\neg z$ literals to x. If some y and $\neg y$ both occur in the conflict graph then immediately after the second one is created, unit propagation will produce the empty clause, so we add to the a single sink vertex labeled $\bot$ and edges from y and $\neg y$ to $\bot$.

Definition 1.13. Given a conflict graph $G = (V, E)$, a *source-sink cut* in G is a set of vertices $U \subset V$ such that

- U contains all sources (decision literals) in G,
- U does not contain the sink node labeled $\bot$, and does not contain both y and $\neg y$.
- U there are no edges from $V - U$ to U in G.

Given such a cut U, let E_U be the set of edges that lead from U to $V - U$. Observe that E_U is a set of edges whose removal eliminates all paths from source nodes to $\bot$. Given such a cut U, we define *conflict clause* C_U to be the clause whose literals are negations of literals at tails of edges in E_U.

Lemma 1.14. *Let G be a conflict graph associated with a leaf node of DPLL search tree for CNF formula F with trail A. Let U be a source-sink cut in G. Then C_U can be derived from F using a resolution derivation of length at most the length of A.*

Proof. Exercise. □

CDCL adds one or more conflict clauses C_U to the formula F at each conflict. There are several potential heuristics for choosing which conflict clause(s) to add. Most commonly, precisely one conflict clause C_U is added and this clause has the additional property that it is *asserting*, which means that adding it will cause unit propagation at some prefix of the trail associated with a smaller decision level, call the *assertion level* of C_U.

If we choose the set U to be the set of decision literals, then the conflict clause C_U will be asserting just one level above the last decision level and adding U will propagate the opposite value for the last decision literal; i.e., adding it and simply using unit propagation will force the same behavior as ordinary backtracking. This allows us to write CDCL iteratively rather than recursively.

More generally, better choices of the conflict clause are asserting at much smaller decision levels. We these choices we can prune off much more than the last decision level, which has the same effect as backtracking over many levels of the tree at once.

Most current CDCL solvers choose a particular conflict clause, called the *1UIP* clause, based on the 1UIP cut which has precisely one literal separating the conflict from the last decision literal and is closest to the conflict among all such cuts, has been shown to be quite effective in practice. They also use properties of the learned clauses during the process to update their heuristics for selection of decision variables. Depending on properties of the learning, e.g., learned clauses that force literals immediately or after a certain number of conflicts, they can also choose to *restart*, which keeps the added conflict clauses but resets the trail A to nil and decision level to 0.

Finally, modern CDCL solvers also apply *learned clause minimization* to the conflict clause before adding it, by removing any literal from the conflict clause C_U that is separated from the decision literals by other literals in C_U; this is equivalent to ensuring that the set U is chosen so that E_U is a *minimal* separating set and hence by Lemma 1.14 can be derived via resolution.

For implementation, it is very important to use the watched literal technique. We therefore obtain the general form of the CDCL algorithm:

Combining all the pieces together we see that the following holds.

Theorem 1.15. *CDCL refutations are resolution refutations.*

From a practical point of view, the number of learned clauses can get quite large so CDCL solvers will also have heuristics for removing learned clauses that are too long or have not been recently used.

Frege systems

These dynamic proof systems are the closest to the formal proof systems of typical introductory logic texts, consisting of a sequence of proof lines that are arbitrary Boolean formulas. These systems were named by Cook and Reckhow after Gottlob Frege who was responsible for many early developments in logic but is now best known for developing a more general system that was demolished by Russell's paradox.

Each *Frege system* $\mathscr{F}$ is given by a finite set of sound axioms and logical inference rules (used as schemas) that together are *implicationally complete*[2].

Cook and Reckhow noted the following:

Theorem 1.16. *Any two Frege systems polynomially simulate each other*

Proof. Because of implicational completeness, any axiom or inference rule in one system has a derivation in the other system and, since the number of these is finite, each such derivation is of constant size. We now simply replace each application of a rule in one system by the substituted version of the proof in the other system. This increases the proof size at most by a constant factor. □

At the level of Frege systems, the tree-like restriction is not meaningful.

[2] Ordinary completeness only requires that the rules should be able to derive any tautology starting from the axioms. Implicational completeness also requires that if we add arbitrary formulas as given formulas, then the system can infer any formula that is logically implied by them

Algorithm 2: CDCL for satisfiability search.

```
1:  function CDCL(F)
2:      Choose two watched literals for each clause of F, if possible.
3:      Set A to nil and conflict graph to empty.
4:      Set decision level to 0.
5:      while true do
6:          while F contains a clause with only 1 watched literal x do
7:              A ← (A, x)
8:              Propagate(x,F)                                  ▷ Unit propagation
9:              Add x and edges to conflict graph
10:         if F has no active clauses then Halt and output satisfying assignment A
11:         if conflict graph has conflicting literals y and ¬y then
12:             if decision level=0 then Halt and output "unsatisfiable".
13:             Use conflict graph to find asserting conflict clause C_U   ▷ Analyze Conflict
14:             Minimize C_U
15:             Add C_U to F and add two watched literals for C_U
16:             Update decision heuristic
17:             if restart chosen then
18:                 Set A to nil and conflict graph to empty.
19:                 Set decision level to 0.
20:             else
21:                 Set decision level to assertion level of C_U, pruning A and the conflict graph.
22:             Update watched literals.
23:         else
24:             Choose unset literal x according to decision heuristic   ▷ Decision literal
25:             A ← (A, x)
26:             Propagate(x,F).
27:             Add source x to conflict graph.
```

Lemma 1.17. *Any Frege proof can be converted to tree-like form in polynomial time*

Given these equivalences, we lump all Frege systems together and view them as one proof system. Frege systems can be used equivalently to prove tautologies or to refute CNF formulas.

Resolution proofs naturally form a restricted form of Frege proof in which the only allowed lines are clauses. (The resolution rule $A \vee x,\ B \vee \neg x \vdash A \vee B$ is a special case of a general rule

$$\text{(Cut)} \qquad \frac{A \vee C \qquad B \vee \neg C}{A \vee B}$$

With clauses, the only way to match the cut rule is if C is a literal.)

Unlike resolution proofs, where each proof line has size at most the number of variables, lines in Frege proofs can be quite large, so the the proof size is the total number of symbols in the proof rather than the number of lines.

Extended Frege proofs

The axiom and inference schemas of Frege proofs have the property that all variables they discuss can take on arbitrary truth values independent of other variables, except as specified by the axioms in the case of refutation systems.

Extended Frege proof systems expand the possibilities by allowing the introduction of new *extension* variables that are defined to be equivalent to entire formulas in other variables. That is, they allow the addition of lines $y \leftrightarrow A$ where A is an arbitrary formula not involving y and y is a new variable that has not appeared previously.

We can apply extension rules with multiple lines When we cannot directly represent $\leftrightarrow$ such a system. In particular, we can add an extension rule to resolution, that lets us define new variables to be equivalent to arbitrary clauses in a way that it is similar to the addition of the gate variables used to convert formulas to CNF. In particular, to set y equivalent to clause $x_1 \vee \cdots \vee x_t$, we add clauses

$$\begin{aligned} &\neg y \vee x_1 \vee \cdots \vee x_t \\ &\neg x_1 \vee y \\ &\cdots \\ &\neg x_t \vee y. \end{aligned}$$

Theorem 1.18. *Extended Frege proofs are all equivalent to resolution with the extension rule.*

Proof. Since resolution is a special case of Frege proofs, one direction is immediate. The idea for the other direction is that using the usual gate

transformation for circuits to CNF formulas using the extension rule, we can define resolution variables so that any line of the extended Frege proof can be expressed as a clause and any inference rule can be derived using the resolution rule on these variables. □

Another way to obtain a proof system based on Frege systems that is equivalent to extended Frege is to add a *substitution rule* to any Frege system.

The substitution rule applies to Frege systems used as proof systems to derive the formula being proved, rather than for refutation. In this forward format, every derived line of the Frege proof is itself a tautology. Add the substitution rule allows any line to be used immediately as an axiom schema. We will not prove the following:

Theorem 1.19. *Extended Frege is equivalent to Frege with the substitution rule added.*

In fact, this equivalence applies even with a very restricted form of the substitution rule, in which the only changes allowed to a formula are to substitute some variables by constants 0/false and 1/true.

Finally, we see another direct connection between Frege and extended Frege proofs.

Theorem 1.20. *There is a constant c such that For any formula F, the* size *of an extended Frege proof of F is at most $c(|F|^2 + L)$ where L is the optimal number of* lines *in a Frege proof of F.*

Proof. Using extension variables, each line of the Frege proof can be represented by a constant-size formula in extended Frege. The $|F|^2$ term is for the final unwinding or extension steps that produces the formula F from these succinct formulas. □

Beyond extended Frege

In principle, we may not need to stop at extended Frege. Though the propositional logic deals with finite numbers of assignment, it may be possible that reasoning about infinite objects can help us to reason about them. We could even use a logical system that covers the vast majority of mathematical reasoning such as *Zermelo-Frankl set theory with the Axiom of Choice* (usually denoted by ZFC) to derive our proofs ... or possibly something even more powerful.

Getting more down to earth, our ability to make use of propositional proofs seems better for simpler systems, and our ability to analyze proof complexity is already challenged for much simpler proof systems. We therefore next consider a natural way to think of a hierarchy of simpler natural proof systems of increasing strength.

Proofs using circuits: $\mathscr{C}$-Frege proofs

Many circuit complexity classes $\mathscr{C}$ are defined in terms of classes of circuits that satisfy some structural property $P_{\mathscr{C}}$ together with a restriction that the size of such circuits is polynomial. Standard classes of this sort in increasing power include

- Clauses
- k-DNF (polynomial-size k-DNF formulas)
- AC^0 (polynomial size constant-depth unbounded fan-in $\wedge, \vee, \neg$-circuits),
- $\mathsf{AC}^0[m]$ (polynomial size constant-depth unbounded fan-in $\wedge, \vee, \neg$, mod m circuits),
- TC^0 (polynomial size constant-depth threshold circuits),
- NC^1 ($O(\log n)$ depth fan-in 2 $\wedge, \vee, \neg$ circuits, which are equivalent to polynomial size Boolean formulas because any polynomial-size Boolean formula can be re-balanced to $O(\log n)$ depth), and
- $\mathsf{P/poly}$ (polynomial-size circuits).

For each such class $\mathscr{C}$, we can define a natural proof system, *$\mathscr{C}$-Frege* proofs, as a dynamical proof system having lines that are circuits satisfying the structural property given by $P_{\mathscr{C}}$ and having axioms and inference rules that are sound and implicationally complete for such circuits.

Using this general framework, we see that:

- Frege $\equiv \mathsf{NC}^1$-Frege
- Extended Frege $\equiv \mathsf{P/poly}$-Frege
- Resolution $\equiv$ Clauses-Frege

The proof system k-DNF-Frege generalizing resolution has been studied under the name $Res(k)$. There has also been extensive study of AC^0-Frege, for which we know a number of strong lower bounds.

Somewhat similar to the situation in circuit complexity, we know very little about the power and limitations of the TC^0-Frege and $\mathsf{AC}^0[m]$-Frege proof systems.

In particular, unlike the situation with circuits, we don't even know much about $\mathsf{AC}^0[p]$-Frege for prime p. These proofs are closely related to algebraic proofs which we consider next.

1.4 Algebraic proof systems

We can use algebra instead of logic to formalize constraints using systems of polynomial equalities.

In particular, we can enforce that a variable x only can take on Boolean values 0 or 1 via the polynomial constraint $x^2 - x = 0$. Similarly, we can enforce a clause constraint such as $C = (x \vee \neg y \vee z)$ via the polynomial constraint $p_C = (1-x)y(1-z) = 0$, or equivalently $y - yz - xy + xyz = 0$.

A key theorem of commutative algebra indicates how we can use algebra to produce proof systems for $CNF\text{-}UNSAT$:

Theorem 1.21 ((Weak) Hilbert's Nullstellensatz)**.** *Let $\mathbb{K}$ be a field and $f_1, \ldots, f_m$ be multivariate polynomials in $\mathbb{K}[x_1, \ldots, x_n]$. Then the system of equations $f_1(x_1, \ldots, x_n = 0, \ldots, f_m(x_1, \ldots, x_n) = 0$ does* not *have a solution in the algebraic closure of $\mathbb{K}$ if and only if there exist polynomials $g_1, \ldots, g_m \in \mathbb{K}[x_1, \ldots, x_n]$ such that $\sum_{i \in [m]} g_i \cdot f_i \equiv 1$.*

In algebraic language, the set of polynomials that are linear combinations of a set of polynomials $f_1, \ldots, f_m \in \mathbb{K}[x_1, \ldots, x_n]$ with coefficients in $[K][x_1, \ldots, x_n]$ is called the *ideal* generated by $f_1, \ldots, f_m$. In other words, it says that no solution exists to this system of equations iff 1 is in the ideal generated by $f_1, \ldots, f_m$.

This theorem allows us to translate a universal statement, the non-existence of solutions, into an existential one, the existence of polynomials, that can be easily checked; The g_i form a proof of the unsatisfiability of the system of equations in the f_i.

As stated in this general form, the Nullstellensatz refers to the algebraic closure of $\mathbb{K}$. However, in our application to propositional logic proofs, among the Q_i we will always have $x_j^2 - x_j$, so any solution must have 0-1 values and hence lie in $\mathbb{K}$ itself.

in addition, these polynomials ensure that the degrees involved are not large. Given a polynomial p, let $multilinearize(p)$ be a polynomial that is a modification of p in which every exponent larger than 1 has been reduced to 1:

Proposition 1.22. *For any polynomial $p \in \mathbb{K}[x_1, \ldots, x_n]$ of total degree at most d, there are polynomials $r_1, \ldots, r_n$ of total degree at most $d-2$ such that $p = multilinearize(p) + \sum_{j \in [n]} r_j(x_j^2 - x_j)$.*

Another way of saying this is that p and $multilinearize(p)$ are equivalent modulo the ideal generated by $x_1^2 - x_1, \ldots, x_n^2 - x_n$ which we denote by I. In particular, because of the ability to multilinearize polynomials, there is never a need for propositional proofs to have any $g_i \cdot f_i$ with degree more than $n+1$.

The degree of algebraic proofs is a natural measure but not the only aspect that needs to be considered in discussing their size. We will say that their *size* is the number of monomials they contain. Finally, there is the impact of the representation size required for each coefficient. (For this last quantity, in the case of fields like $\mathbb{R}$ or $\mathbb{Q}$, we assume that all polynomials have *integer* coefficients.) The *bitsize* of a proof is the total number of bits required to represent it in this standard representation.

In our standard monomial representation, even simple constructs like large clauses such as $C = x_1 \vee \cdots \vee x_n$ require exponential size because $p_C = \prod_{j\in[n]}(1-x_j)$ has 2^n monomials. To get around this annoying problem, for each variable in our CNF formula, x, we sometimes add a dual variable $\overline{x}$ together with the polynomial constraint $x + \overline{x} - 1 = 0$. In that case, we replace each $1-x$ by $\overline{x}$ in our translations and any We generally assume that we have these dual variables (also known as paired variables) and include all of the dual constraints to the ideal I' and work modulo the resulting ideal I' (which allows us to replace $x + \overline{x}$ by 1).

Nullstellensatz proofs

For a sequence of polynomials $f_1, \ldots, f_m$ in variables $z_1, \ldots, z_n$ over field $\mathbb{F}$, a *Nullstellensatz refutation* of the system of polynomial equations $\tilde{F} = (f_1 = 0, \ldots, f_m = 0)$ is an explicit list of polynomials $g_1, \ldots, g_m$ over $\mathbb{F}$ written as sums of monomials such that

$$\sum_{i\in[m]} f_i \cdot g_i \equiv 1. \tag{1.1}$$

where $\equiv$ denotes equality of polynomials rather than values. The *degree* of a Nullstellensatz refutation is $\max_{i\in[m]} \deg(g_i \cdot f_i)$.

For a CNF formula $F = \wedge_{i\in[m]} C_i$ in variables $x_1, \ldots, x_n$ and field $\mathbb{F}$, we define a Nullstellensatz refutation of F to be a Nullstellensatz refutation of the polynomial system $\tilde{F} = (p_{C_1} = 0, \ldots, p_{C_m} = 0, x_1^2 - x_1 = 0, \ldots, x_n^2 - x_n = 0)$. We define $\mathrm{NSdeg}_{\mathbb{F}}(F)$ to be the minimum degree of any Nullstellensatz refutation of F.

Since there are only $\binom{n}{d}$ multilinear monomials of degree d in n variables, a Nullstellensatz proof of degree d can have at most $(m+n)\cdot\binom{n}{\leq d}$ coefficients which is $m \cdot n^{O(d)}$. For example, any constant-degree Nullstellensatz proof over a finite field is automatically polynomial size.

Finding Nullstellensatz refutations If we include an indeterminate for each coefficient of degree up to $d - \deg(f_i)$ in each g_i, then (Eq. (1.1)) yields a system of linear equations with coefficients determined by the coefficients of the p_C for $C \in F$, one equation per monomial of degree up to d. If a solution exists, then we can find it using linear algebra on this system, which has size at most $m \cdot n^{O(d)}$. Therefore, if a degree d Nullstellensatz refutation of F exists, it can be found in $m^3 \cdot n^{O(d)}$ field operations, which is polynomial if the degree d is a constant.

Proposition 1.23. *Prove that for any tree resolution refutation and any field $\mathbb{F}$ there is a Nullstellensatz refutation over $\mathbb{F}$ whose degree is at most the height of the tree resolution refutation.*

Proof. Exercise. □

Though Proposition 1.23 is independent of the choice of field, Nullstellensatz over fields of different characteristic yield incomparable proof systems.

Polynomial Calculus and PCR proofs

Nullstellensatz proofs are not dynamic proofs – the g_i are chosen at once. *Polynomial calculus (PC)* is a corresponding dynamic proof system based on the same initial polynomials and algebraic idea as Nullstellensatz proofs: to prove unsatisfiability it suffices to prove that 1 is in the ideal generated by the initial polynomials. In particular, a polynomial calculus derivation of $f = 0$ from $\tilde{F} = (f_1 = 0, \ldots, f_m = 0)$ over field $\mathbb{F}$ is the sequence of polynomials $f_1, \ldots, f_m, f_{m+1}, \ldots, f_t$ such that for $i > m$, either

- (Linear combination) $f_i = a \cdot f_j + b \cdot f_{j'}$ for some $a, b \in \mathbb{F}$ and $j, j' < i$, or
- (Multiplication by variable) $f_i = x_k \cdot f_j$ for $k \in [n]$ and $j < i$.

Such a derivation is a refutation of $\tilde{F}$ iff $f_t = 1$. The degree of the derivation is $\max_i \deg(f_i)$. We write $f_1, \ldots, f_m \vdash_d f$ if there is a derivation of f from $f_1, \ldots, f_m$ with degree at most d.

For a CNF formula F in $x_1, \ldots, x_n$, a *PC refutation of* F is a polynomial calculus refutation of the set of polynomials $\mathscr{E}(F) = \{p_C \mid C \in F\} \cup \{x_i^2 - x_i \mid i \in [n]\}$. Often, rather than including the polynomials $x_i^2 - x_i$ as separate input polynomials we assume that the inference rules are being taken modulo the ideal I.

We denote the minimal degree of any PC refutation of F over field $\mathbb{F}$ as $\deg_{\mathbb{F}}(F)$.

As noted above, we use *PCR* (which stands for Polynomial Calculus with Resolution) to denote the refutation system for CNF formulas in which we add the dual variables $\overline{x}_i$, initial polynomials $x_i + \overline{x}_i - 1$ for $i \in [n]$ and using the algebraic translation $\overline{p}_C$ instead of p_C for clauses yielding the translation which we can $\mathscr{E}'(F)$. In PCR we generally also work modulo the ideal I' defined by all $x_i^2 - x_i$ and $1 - x_i - \overline{x}_i$. (Note that $\overline{x}_i^2 - \overline{x}_i$ is also immediately derived and that the degree required for PCR proofs is identical to that for PC.)

The following theorem gives a good explanation for the PCR terminology.

Theorem 1.24. **PCR**$_{\mathbb{F}}$ *polynomially simulates resolution.*

Proof. We already have $\overline{p}_C$ for each input clause C. We show that we can also efficiently derive $\overline{p}_D$ for each derived clause D in the resolution refutation. Suppose $A \vee B$ is the resolvent of $A \vee x$ and $B \vee \neg x$ and that inductively that we already have $\overline{p}_{A \vee x}$ and $\overline{p}_{B \vee \neg x}$. Let C be the set of common literals in A and B. Then by definition $\overline{p}_{A \vee x} = \overline{p}_{A - C} \cdot \overline{p}_C \cdot x$ Therefore in $|B - C|$ multiplication steps we obtain

$$q_0 = \overline{p}_{B-C} \cdot \overline{p}_{A \vee x} = \overline{p}_{A-C} \cdot \overline{p}_{B-C} \cdot \overline{p}_C \cdot x = \overline{p}_{A \vee B} \cdot x.$$

Similarly, $\overline{p}_{B\vee\neg x} = \overline{p}_{B-C}\cdot\overline{p}_C\cdot\overline{x}$ and after $|A-C|$ multiplication steps we obtain

$$q_1 = \overline{p}_{A-C}\cdot\overline{p}_{B\vee\neg x} = \overline{p}_{A-C}\cdot\overline{p}_{B-C}\cdot\overline{p}_C\cdot\overline{x} = \overline{p}_{A\vee B}\cdot\overline{x}.$$

Then $q_0+q_1 = \overline{p}_{A\vee B}\cdot(x+\overline{x})$. Modulo the ideal I', $x+\overline{x}=1$ so this is just $\overline{p}_{A\vee B}$ as required. In total this is at most n $\mathbf{PCR}_{\mathbb{F}}$ steps and each line (except for q_0+q_1) is of the same size as the corresponding clause. □

Finding PC and PCR proofs PC and PCR proofs of small degree d also can be shown to have derivations of size at most $\binom{n}{d}^{O(1)}$ or $n^{O(d)}$. Here is an argument for PC; the one for PCR is similar: For each $d' \le d$ we construct a basis of the vector space

$$V_{d'} = \{f \mid f \text{ is multilinear and } f_1,\ldots f_m \vdash_{d'} f\}.$$

In particular $V_0 = 0$ and the basis is empty. Also, $V_{d'}$ has dimension at most $\binom{n}{\le d'}$, the number of multilinear monomials of degree at most d'. Suppose that we have a basis $B_{d'}$ for $V_{d'}$ for $d' < d$. To compute the basis $B_{d'+1}$ for $V_{d'+1}$:

- View each element $p \in B_{d'}$ as an element of $V_{d'+1}$.
- For each $p \in B_{d'}$ and each x_i for $i \in [n]$, add the polynomial multilinear$(x_i\cdot p)$.
- For any input polynomial p add multilinear(p) if it has degree exactly $d'+1$.
- Use Gaussian elimination to reduce the resulting set of polynomials (viewed as vectors) to a basis $B_{d'+1}$.

For each d', we can check to see if the polynomial 1 is spanned by the basis $B_{d'}$ and stop only when that is found.

In practice, there is a more refined algorithm called the *Groebner (or Gröbner) Basis Algorithm* that is used. There are a number of well-known Groebner basis algorithm software packages out there but most of this software is designed for the general case when the equations $x^2-x=0$ are not present – in that general case, the degree required can be as large as double exponential in n. The general idea is to iteratively convert the polynomials defining the ideal into a minimal collection of *rewrite rules* that replace the *leading term* of a (monic) polynomial (the largest term in a total order on terms based on a lexicographic order for the exponent vectors of its monomials) by a linear combination of smaller terms. The degree and number of these rewrite rules can be much larger the the input polynomials.

Beyond PCR

Since the variables we consider are Boolean, multiplication in PC and PCR can be viewed as a Boolean AND. Over a finite field such as $\mathbb{F}_p$, a polynomial

can be viewed as a depth 2 unbounded fan-in circuit with AND gates feeding into a MOD_p gate and hence a special case of $AC[\oplus p]$-Frege.

Another direction that takes us beyond our formal definition of proof system is to consider proofs based on Hilbert's Nullstellensatz involving polynomials represented by algebraic circuits rather than as explicit sums of monomials. The difficulty is that is not known how to check in deterministic polynomial time if a polynomial given by a circuit is equal to the polynomial 1. However, this is something that can be verified up to arbitrarily small probability of error by randomized algorithms running in polynomial time, using the Schwartz-Zippel Lemma, yield a *randomly-checkable proof system*. The natural randomly-checkable proof system that best captures this idea is known as the *Ideal Proof System (IPS)*.

1.5 Semi-algebraic proof systems

While the algebraic proof systems are defined over any field and based on polynomial equalities, all *semi-algebraic* proof systems are defined only over $\mathbb{R}$ and are based on polynomial *inequalities*.

Semi-algebraic proof inequalities always include $x_i \geq 0$ and $1 - x_i \geq 0$ (equivalently $-x_i \geq -1$) for $i \in [n]$. The standard form of the semi-algebraic translation of a clause $C = (\vee_{i\in P} x_i) \vee (\vee_{i\in N} \neg x_i)$, is the inequality $\ell_C \geq 0$ where

$$\ell_C = \sum_{i\in P} x_i + \sum_{i\in N} (1 - x_i) - 1.$$

For some semi-algebraic proof systems we prefer to move the constant term to the right and express this as

$$\sum_{i\in P} x_i - \sum_{i\in N} x_i \geq 1 - |N|,$$

an inequality we denote as L_C.

We sometimes also discuss equalities in semi-algebraic proof systems; each equality is really a pair of inequalities.

In practice, as in algebraic systems, it is often convenient to add dual variables $\overline{x}_i$, together with constraints that $\overline{x}_i + x_i = 1$; that is,

$$x_i + \overline{x}_i \geq 1$$
$$-x_i - \overline{x}_i \geq -1$$

for each variable. In this case we can translate the clause C above as

$$\sum_{i\in P} x_i + \sum_{i\in N} \overline{x}_i \geq 1.$$

which we denote by L'_C.

*Cutting planes (**CP**) proofs*

Cutting planes is a dynamic proof system whose lines are integer linear inequalities of the form $\sum_i a_i \cdot x_i \geq b$ where $a_1, \ldots, a_n, b$ are integers. Cutting planes proofs were originally developed by Gomory and Chvatal for general inference for integer programming and not just for propositional logic. The rules we discuss are fully general but our focus will be on their use for propositional logic.

A *cutting planes derivation* of a linear inequality L from a system of integer linear inequalities $Ax \geq b$ for $A \in \mathbb{Z}^{m \times n}$, is a sequence of linear inequalities $L_1, \ldots, L_t$ where L_i is $\sum_{j \in [n]} A_{ij}\ x_j \geq b_i$ for $i \in [m]$ and each L_i for $i > m$ follows from previous lines by one of the following inference rules:

$$\text{(Addition)} \qquad \frac{a_1\, x_1 + \cdots + a_n\, x_n \geq c \qquad b_1\, x_1 + \cdots + b_n\, x_n \geq d}{(a_1 + b_1)\, x_1 + \cdots + (a_n + b_n)\, x_n \geq c + d}$$

$$\text{(Multiplication)} \qquad \frac{a_1\, x_1 + \cdots + a_n\, x_n \geq b}{ca_1\, x_1 + \cdots + ca_n\, x_n \geq cb} \quad \text{for any positive integer } c$$

$$\text{(Division)} \qquad \frac{ca_1\, x_1 + \cdots + ca_n\, x_n \geq b}{a_1\, x_1 + \cdots + a_n\, x_n \geq \lceil b/c \rceil} \quad \text{for any positive integer } c$$

Such a derivation is a refutation of $Ax \geq b$ iff the last line L_S is $0 \geq 1$. The multiplication and addition rules can be combined in a single rule that allows one to take any positive linear combination of two inequalities. That combined rule is sound over the reals.

For a CNF formula $F = \wedge_{i \in [m]} C_i$ on variables $x_1, \ldots, x_n$, a **CP** refutation of F is cutting planes refutation of the system of constraints

$$L_{C_1}, \ldots, L_{C_m}, x_1 \geq 0, \ldots, x_n \geq 0, 1 - x_1 \geq 0, \ldots 1 - x_n \geq 0.$$

This refutation is a **CP**$*$ refutation if every coefficient in the inequalities in the refutation has absolute value at most polynomial in the size of the input.

Polytope view Any set of inequalities defines a polyhedron over the reals. Because we have the bounding constraints that each variable x_i is in $[0, 1]$, the input inequalities actually define a polytope P (a bounded polyhedron) which has rational vertices since the coefficients are all integers. Cutting planes rules allow us to derive constraints corresponding to the convex hull of the set of integer points in P, known as the *integer hull* of P. A refutation proves that the integer hull is empty.

Each inequality corresponds to a halfspace and two inequalities describe an intersection of halfspaces. Unless one inequality implies the other over $\mathbb{R}$, this will have an *apex* defined by the intersection of their defining hyperplanes. The positive linear combination rule derives a new halfspace whose defining hyperplane contains that apex and, by soundness over the reals, contains the intersection of the two halfspaces.

The only progress in moving from P to its integer hull happens during applications of the division rule. The application of this rule essentially shifts the constraint until its defining hyperplane intersects with some integer point, cutting off part of P in the process, hence the name for the proof system.

Theorem 1.25. *Cutting planes (**CP**$*$) polynomially simulates resolution.*

Proof. The proof shows inductively that for any clause C of a resolution refutation, **CP**$*$ can efficiently derive the corresponding inequality L_C. This is already the case for the input clauses by definition; for convenience, we think of the translation of a clause $C = \bigvee_{i\in P} x_i \vee \bigvee_{i\in N} \neg x_i$ as

$$\sum_{i\in P} x_i + \sum_{i\in N}(1-x_i) \geq 1.$$

Suppose that we have already derived inequalities $L_{A\vee x}$ and $L_{B\vee\neg x}$ and that C is the set of common literals appearing in both A and B. That is, we have by induction,

$$\begin{aligned} x + \sum_{i\in P_A} x_i + \sum_{i\in N_A}(1-x_i) &\geq 1 \\ 1-x + \sum_{i\in P_B} x_i + \sum_{i\in N_B}(1-x_i) &\geq 1 \end{aligned}$$

Adding these we obtain

$$1 + \sum_{i\in P_{A-C}\cup P_{B-C}} x_i + \sum_{i\in N_{A-C}\cup N_{B-C}} (1-x_i) + 2\cdot\Big(\sum_{i\in P_C} x_i + \sum_{i\in N_C}(1-x_i)\Big) \geq 2.$$

Then, subtracting 1 from both sides, and for all $i \in P_{A-C} \cup P_{B-C}$ adding $x_i \geq 0$ and all $j \in N_A \cup N_B$, adding $1 - x_j \geq 0$ yields

$$2\cdot\Big(\sum_{i\in P_{A\cup B}} x_i + \sum_{i\in N_{A\cup B}} (1-x_i)\Big) \geq 1;$$

so, dividing by 2 and rounding up by the division rule we obtain

$$\sum_{i\in P_{A\cup B}} x_i + \sum_{i\in N_{A\cup B}} (1-x_i) \geq 1,$$

which is precisely $L_{A\vee B}$ as required and takes $O(|A \vee B|)$ steps. □

While working over $\mathbb{R}$ might seem to allow the possibility that arbitrarily large coefficients might be need, the following shows that only $O(n \log n)$ bit coefficients are required.

Proposition 1.26 (Muroga)**.** *Any linear threshold formula $a_1 \cdot x_1 + \cdots + a_n \cdot x_n \geq b$ over Boolean variables $x_1, \ldots, x_n$ for $a_1, \ldots, a_n, b \in \mathbb{R}$ is equivalent to a linear threshold formula $a'_1 \cdot x_1 + \cdots + a'_n \cdot x_n \geq b'$ in which all $a'_1, \ldots, a'_n, b \in \mathbb{Z}$ and each has absolute value at most $2^{(n\log_2 n)/2-1}$.*

This implies that every line of a cutting planes refutation can be expressed by a threshold formula using only $O(n^2 \log n)$ bits. It also implies the following simulation.

Corollary 1.27. *TC^0-Frege polynomially simulates* **CP**.

In particular since $\mathsf{TC}^0 \subseteq \mathsf{NC}^1$, Frege also polynomially simulates **CP**.

Buss and Clote showed that as a propositional proof system in the definition of the division rule, it suffices to restrict oneself to the case that $c = 2$.

Pseudo-Boolean solvers: propositional cutting planes in practice In practice, there are a number of solvers, called *pseudo-Boolean (pB)* solvers, that build on the ideas of CDCL solvers but apply them to integer linear inequalities for Boolean variables. These solvers generally are used on the dual variable translations for input clauses which is more convenient and automatically replaces all instances of $x + \overline{x}$ on the left with -1 on the right.

The analogue of unit propagation used by these solvers is

- (Propagation) If $a_1\, x_1 + \cdots + a_n\, x_n \geq b$ and $\sum_{i \neq j} a_i < b$ then infer $x_i = 1$.

pB algorithms generally maintain a sorted list of the (positive) coefficients of each inequality (together with the total weight of its coefficients) which makes it easy to find literals to propagate.

This allows pB solvers to match DPLL-style solvers which are just branching on variable values. However, the interesting part of the development of pB solvers is how they derive new constraints when conflicts are found in the constraint-learning process of CDCL. There are many tricky issues in conflict analysis and in most solvers the production of an asserting conflict constraint is an iterative process of multiple rounds of constraint refinement.

Many of these pB solvers implement only weak fragments of cutting planes rules; they rarely implement something equivalent to the full division rule or full positive linear combination.

They typically only allow positive linear combination if the result eliminates a variable x and its dual variable $\overline{x}$ by cancelling x in one against $\overline{x}$ in the other. This restriction is called *cancelling linear combination*. This restriction to cancelling linear combination is the reason that learned constraints from direct translations of clauses stay as translations of clauses. In that case, the coefficients are always 1 so there is no use for the division rule.

Instead of the full division rule, many solvers use the rules

$$\text{(Saturation)} \qquad \frac{a_1\, x_1 + \cdots + a_n\, x_n \geq b}{\min(a_1, b)\, x_1 + \cdots + \min(a_n, b)\, x_n \geq b}$$

$$\text{(Weakening)} \qquad \frac{a_1\, x_1 + \cdots + a_j\, x_j + \cdots + a_n\, x_n \geq b}{a_1\, x_1 + \cdots + (a_j - 1)\, x_j + \cdots + a_n\, x_n \geq b - 1}$$

Saturation is obviously sound. Weakening is sound because $-x_j \geq -1$, and can enable saturation to be more effective.

Solvers that do include the full division rule implement it without assumption as

$$\text{(Division')} \quad \frac{a_1 x_1 + \cdots + a_n x_n \geq b}{\lceil a_1/c \rceil x_1 + \cdots + \lceil a_n/c \rceil x_n \geq \lceil b/c \rceil} \quad \text{for any positive integer } c$$

Since we have $x_i \geq 0$ for all i, we can see that this follows from the original division rule by first rounding up each a_i to the next multiple of c. A drawback is that, unlike the original division rule in which the use of the rule is signalled by the common factor c, this new rule can result in a substantial loss of information from the original constraint. In practice this version of the rule is also used to avoid coefficient explosion because the least common multiples required to use cancelling linear combination can result in significant coefficient growth over time.

*Sherali-Adams (**SA**) proofs*

This proof system is the direct analogue for polynomial inequalities over the reals of the algebraic Nullstellensatz proof system for inequalities over arbitrary fields. Like Nullstellensatz, it is a static proof system. and represents its polynomials as explicit sums of (generalized) monomials.

Given a system of polynomial inequalities and equalities $(h_1 \geq 0, \ldots, h_m \geq 0; f_1 = 0, \ldots, f_m = 0)$ over the reals in variables $x_1, \ldots, x_n$, a *Sherali-Adams derivation* of a polynomial inequality $h \geq 0$ is an explicit list of real polynomials $g_0, \ldots, g_m; e_1, \ldots, e_{m'+n}$ such that

$$g_0 + \sum_{i=1}^{m} g_i \cdot h_i + \sum_{i \in [m']} e_i \cdot f_i + \sum_{i=1}^{n} e_{m'+i}(x_i^2 - x_i) = h. \tag{1.2}$$

and for each $i \in [0, m]$, g_i is a positive linear combination of generalized monomials of the form $\prod_{i \in P} x_i \cdot \prod_{j \in N}(1 - x_j)$ for disjoint subsets $P, N \subseteq [n]$. (These generalized monomials are called *non-negative juntas* and naturally would correspond to ordinary monomials if we had dual variables.)

Note that for the equalities, we have no constraint on the form of their coefficient polynomials. The *degree* of a Sherali-Adams derivation is the maximum of $\max_i \deg(g_i \cdot h_i)$ and $\max_i \deg(e_i \cdot f_i)$.

We often find it more convenient to rephrase Eq. (1.2) as

$$g_0 + \sum_{i=1}^{m} g_i \cdot h_i + \sum_{i \in [m']} e_i \cdot f_i \equiv_I h. \tag{1.3}$$

where I is the ideal generated by $x_1^2 - x_1, \ldots, x_n^2 - x_n$ since we never require terms involving ideal I of degree larger than the rest.

A *Sherali-Adams (**SA**) refutation* of a CNF formula $F = \bigwedge_{i\in[m]} C_i$ is derivation of -1 from the system of polynomial inequalities $(\ell_{C_1} \geq 0, \ldots, \ell_{C_m} \geq 0;)$ We write $\mathrm{SAdeg}(F)$ for the minimum degree of any **SA** refutation of F.

As with the Nullstellensatz proof system we often want to think of terms in a Sherali-Adams proof as being computed modulo the ideal I generated by the polynomials $x_1^2 - x_1, \ldots, x_n^2 - x_n$.

Recall our previous translation which took each clause C and translated it as a polynomial equality stating that a particular non-negative junta $p_C = 0$.

Proposition 1.28. *For any clause C in variables $x_1, \ldots, x_n$,*

*(a) there are **SA** derivations of p_C and $-p_C$ of degree $|C|$ and size $O(|C|^2)$ from $(\ell_C \geq 0;)$ and*

*(b) there are **SA** derivations of $\ell_C \geq 0$ of degree $|C|$ and size $O(|C|^2)$ from $(; p_C = 0)$.*

Proof. Let $C = \bigvee_{i\in P} x_i \vee \bigvee_{i\in N} x_i$; then $p_C = \prod_{i\in P}(1-x_i)\cdot\prod_{i\in N} x_i$. For part (a), the inequality $p_C \geq 0$ is immediate since we can set $g_0 = p_C$ and every other polynomial to 0. The derivation of inequality $-p_C \geq 0$ is an easy exercise.

For part (b) we prove by induction on the length of C that $\ell_C = -p_C + J_C$ where J_C is a positive linear combination of non-negative juntas of degree at most $|C|$. For the base case, observe that for variable x, $\ell_X = x - 1 = -(1-x) = -p_x$ and $\ell_{\neg x} = (1-x) - 1 = -x = -p_{\neg x}$. Assume that we have $\ell_C = -p_C + J_C$. Observe that $\ell_C + 1$ is itself a positive sum of non-negative juntas. Then, for any expression y,

$$\begin{aligned}\ell_C + y &= \ell_C(1-y) + (\ell_C+1)y \\ &= (-p_C + J_C)(1-y) + (\ell_C+1)y \quad \text{by the induction hypothesis} \\ &= -p_C(1-y) + J_C(1-y) + (\ell_C+1)y\end{aligned}$$

Plugging in $y = x$ we get

$$\begin{aligned}\ell_{C\vee x} = \ell_C + x &= -p_C(1-x) + J_C(1-x) + (\ell_C+1)x \\ &= -p_{C\vee x} + J_C(1-x) + (\ell_C+1)x\end{aligned}$$

and plugging in $y = 1-x$ we get

$$\begin{aligned}\ell_{C\vee\neg x} = \ell_C + (1-x) &= -p_C x + J_C x + (\ell_C+1)(1-x) \\ &= -p_{C\vee\neg x} + J_C x + (\ell_C+1)(1-x)\end{aligned}$$

which give the required expressions. □

By first applying the above proposition to each of the clauses of F and splitting every coefficient polynomial of a Nullstellensatz refutation over the reals (with flipped sign) into two pieces depending on whether $p_C \geq 0$ or $-p_C \geq 0$ should be used, we obtain the following simulation.

Theorem 1.29. *For any CNF formula F, with a Nullstellensatz refutation of degree d and size S over $\mathbb{R}$, there is a Sherali-Adams refutation of F of degree at most d and size at most $O(S + |F| \cdot w(F))$.*

We also can derive the following theorem whose proof is an exercise.

Theorem 1.30. *For any CNF formula F, with a resolution refutation of width w and size S, there is a Sherali-Adams refutation of F using dual variables of degree at most $w + 1$ and size polynomial in w and S.*

Proof Sketch. The general idea is to argue inductively that the negative of the dual variable monomial associated with each derived clause can be expressed as via an SA derivation of small degree and size. We can do this for each input clause in F using Proposition 1.28 to derive $-\overline{p}_C$ in degree $w(F)$. It remains to argue that we can do so for each resolution inference. At the end of the proof, the monomial corresponding to the empty clause is 1, so the result will be a derivation of -1 which is a contradiction. The details are left as an exercise. □

Sherali-Adams proofs are closely related to families of linear programming relaxations of 01-programs of increasing numbers of variables and constraints that provide increasingly tight approximations of the integer (01) convex hull. In particular, the variables that correspond to degree d Sherali-Adams derivations are in 1-1 correspondence with the non-negative juntas of degree at most d. Sherali and Adams' original definitions of these derivations were based on this linear programming formulation.

Finding Sherali-Adams proofs The size of any Sherali-Adams proof of degree at most d is polynomial in $\binom{n}{\leq d}$ or $n^{O(d)}$ as is the size of the linear programming formulation of Sherali-Adams. Since linear programming is polynomial-time solvable, **SA** proofs of degree d may be found in time polynomial in $n^{O(d)}$.

*Sum of Squares (**SoS**) proofs*

Sum of Squares (**SoS**) proofs are a generalization of Sherali-Adams proofs that allows the use of other polynomials that we know must be non-negative over the reals beyond the input constraints and the non-negative juntas: sums of squares of polynomials.

Given a system of constraints defined by polynomial inequalities and equalities

$$(h_1 \geq 0, \ldots, h_m \geq 0; f_1 = 0, \ldots, f_m = 0)$$

over the reals in variables $x_1, \ldots, x_n$, a *sum-of-squares (**SoS**) derivation* of a polynomial inequality $h \geq 0$ is an explicit list of real polynomials

$q_0, \ldots, q_m; r_1, \ldots, r_{m'}$ such that

$$q_0 + \sum_{i=1}^{m} q_i \cdot h_i + \sum_{i \in [m']} r_i \cdot f_i = h. \tag{1.4}$$

and for each $i \in [0, m]$, q_i is a sum of squares of polynomials.

If we consider the subset $K \subseteq \mathbb{R}^n$ defined by the system of constraints then *Eq*. (1.4) is a proof that the polynomial h is always non-negative on K. The question of whether such a proof always exists for a polynomial h that is non-negative on K, was open for a long time and the answer was shown to be negative in general[3] by Motzkin, but it is true in the special case that the set K is bounded.

We will focus on the case that the set K is a subset of the Boolean hypercube $\{0,1\}^n$ by always including the Boolean polynomials $x_1^2 - x_1, \ldots, x_n^2 - x_n$ among the polynomials $f_1, \ldots, f_{m'}$, which certainly guarantees that K will be bounded. As was the case for **SA** we will assume that the proof is done mod the ideal I generated by these Boolean polynomials[4] so the proof that $h \geq 0$ can be written simply as

$$q_0 + \sum_{i=1}^{m} q_i \cdot h_i \equiv_I h. \tag{1.5}$$

where all the q_i are sums of squares, since the terms in the ideal I never need to have larger degree than the rest. When we describe these proofs we will also use the fact that the sum of **SoS** derivations of h and h' is an **SoS** derivation of $h + h'$.

In particular, we will be interested in whether or not K is empty, which we can certify by deriving the conclusion that the polynomial -1 is positive on K.

In particular, a *sum-of-squares (**SoS**) refutation* of a CNF formula $F = \bigwedge_{i \in [m]} C_i$ is an explicit list of real polynomials $q_0, \ldots, q_m$ such that

$$q_0 + \sum_{i \in [m]} q_i \cdot \ell_{C_i} \equiv_I -1 \tag{1.6}$$

and each of $q_0, \ldots, q_m$ is a sum of squares of polynomials.

We now see that Sherali-Adams can be viewed as a special case of **SoS** proofs up to a doubling of the degree.

Proposition 1.31. *Every non-negative junta g of degree d can be expressed as $q + \sum_{j \in [n]} r_j \cdot (x_j^2 - x_j)$ where q is a square and $\deg(q)$ and $\max_j(\deg(r_j)+2)$ are at most $2d$.*

Proof. Since $x = x^2 - 1 \cdot (x^2 - x)$ and $(1-x) = (1-x)^2 - 1 \cdot (x^2 - x)$, the statement is true for non-negative juntas of degree 1. To obtain the result for larger d we simply multiply the expressions, yielding degree $2d$. □

[3] There are two pieces missing in Eq. (1.4) that are required for the general condition. We have to allow sums of *products* of the $h_1, \ldots, h_m$ (which are also non-negative on K) and we have to allow a representation of $q \cdot h$ for some non-zero sum of squares q and not just of h itself.

[4] Working modulo the ideal I is equivalent to being able to replace a polynomial p by multilinear(p).

Since Nullstellensatz, and hence Sherali-Adams, has refutations of degree at most n, we immediately obtain the following:

Corollary 1.32. *Every CNF formula in n variables has an* **SoS** *refutation of degree at most $2n$.*

The following theorem due to Berkholz was quite surprizing since it shows that the dynamic proof system of polynomial calculus over the reals can be efficiently simulated by a purely static system.

Theorem 1.33. *Every $\mathbf{PC}_{\mathbb{R}}$ derivation $f_1, \ldots, f_m; f_{m+1}, \ldots, f_T$ in degree $\leq d$, size $\leq S$, and coefficient bitsize $\leq B$, can be simulated by an* **SoS** *derivation of $-f_T^2$ from $(f_1 = 0, \ldots, f_m = 0; 1 \geq 0)$ of degree $\leq 2d$, size $\leq S^{O(1)}$ and bitsize $B^{O(1)}$.*

Proof. Let $f_1, \ldots, f_m; f_{m+1} \ldots, f_t$ be $\mathbf{PC}_{\mathbb{R}}$ derivation and assume that every coefficient c appearing in it satisfies $1/K \leq 4c^2 \leq K$. The required **SoS** derivation is immediate from the following claim.

Claim 1.34. *There is a sequence of polynomials $q_{m+1}, \ldots, q_T$ of degree at most d such that for every $t \leq T$, there is an expression $\mathscr{E}_t$ of the form*

$$\mathscr{E}_t = \sum_{i=1}^{m} -(c_i f_i) \cdot f_i + \sum_{i=m+1}^{t} c_i q_i^2 \equiv_I -f_t^2.$$

such that $1/K^t \leq c_i \leq K^t$ for each i and every non-zero coefficient of each q_i is at most $2c^2$.

We prove this claim by induction on t.
CASE $t \leq m$: Setting $c_t = 1$ and $c_i = 0$ for all other $i \in [m]$ gives the required expression.

CASE $f_t = x_j f_{t'}$ FOR $t' < t$: Since $\deg(f_t) \leq d$, we have $\deg(f_{t'}) \leq d-1$ and $\mathscr{E}_{t'} \equiv -f_{t'}^2 \pmod{I}$ by the inductive hypothesis. We set $q_t = (1-x_j) f_{t'}$ which has degree at most d and coefficient size at most twice that of $f_{t'}$. Then

$$q_t^2 = (1-x_j)^2 f_t^2 = (1 - 2x_j + x_j^2)\, f_{t'}^2 \equiv_I (1-x_j^2) f_{t'}^2 = f_{t'}^2 - f_t^2.$$

Therefore setting $\mathscr{E}_t = \mathscr{E}_{t'} + q_t^2$ gives us the desired expression.

CASE $f_t = a\, f_{t'} + b\, f_{t''}$ FOR $t', t'' < t$: By induction we have $\mathscr{E}_{t'} \equiv_I -f_{t'}^2$ and $\mathscr{E}_{t''} \equiv_I -f_{t''}^2$.
We set $q_t = a\, f_{t'} - b\, f_{t''}$. Now q_t has degree at most d, and its coefficient size is at most $2c^2$. Then

$$f_t^2 = a^2\, f_{t'}^2 + b^2\, f_{t''}^2 + 2ab\, f_{t'} f_{t''}$$
$$q_t^2 = a^2\, f_{t'}^2 + b^2\, f_{t''}^2 - 2ab\, f_{t'} f_{t''}$$

so

$$f_t^2 + q_t^2 = 2a^2\, f_{t'}^2 + 2b^2\, f_{t''}^2 \equiv_I -2a^2\, \mathscr{E}_{t'} - 2b^2\, \mathscr{E}_{t''}.$$

Now defining $\mathscr{E}_t = 2a^2\, \mathscr{E}_{t'} + 2b^2 \mathscr{E}_{t''} + q_t^2$ we see that $\mathscr{E}_t \equiv_I -f_t^2$ as required.

The polynomials used in the expressions for $\mathscr{E}_{t'}$ and $\mathscr{E}_{t''}$ are part of the common set shared by all these expressions, so the only remaining part is to verify the bounds on the c_i. They could have grown by a factor of at most $2a^2 + 2b^2$ which is at most $4c^2 \leq K$ by assumption. □

Corollary 1.35. **SoS** *polynomially simulates* $\mathbf{PC}_{\mathbb{R}}$.

Proof. Since a $\mathbf{PC}_{\mathbb{R}}$ refutation of a CNF formula $F = \wedge_{i\in[m]} C_i$ derives the polynomial 1, the simulation of Theorem 1.33 yields a derivation of -1 from the initial polynomial equations $p_{C_1} = 0, \ldots, p_{C_m} = 0$. We are almost done, except that in **SoS** we typically assume that we begin with $\ell_{C_1} \geq 0, \ldots, \ell_{C_m} \geq 0$. To complete the argument we simply replace each p_{C_i} by the expression given in Proposition 1.28. (We can convert this expression to **SoS** without increasing the degree.) □

The relationship between **CP** and **SoS** is unclear.

SoS proofs are closely related to families of semidefinite- programming relaxations of 01-programs of increasing numbers of variables and constraints that provide increasingly tight approximations of the integer (01) convex hull. These variables for degree d correspond to monomials of degree at most d. The **SoS** formulation is essentially by Parillo; Lasserre independently gave a dual formulation in terms of constraints on these lifted variables. We will discuss much more about the interpretations of the values of these variables.

Finding **SoS** *proofs* The size of any **SoS** proof of degree at most d is polynomial in $\binom{n}{\leq d}$ or $n^{O(d)}$ as is the size of its semi-definition programming formulation. Since semi-definite programming is polynomial-time solvable, **SoS** proofs of degree d may be found in time polynomial in $n^{O(d)}$.

Positivstellensatz proofs

Positivstellensatz proofs generalize **SoS** by using the fact that the product of two non-negative quantities is itself non-negative. Like **SoS**, **SA**, and Nullstellensatz proofs, these proofs are static.

Given a system of constraints defined by polynomial inequalities and equalities

$$(h_1 \geq 0, \ldots, h_m \geq 0; f_1 = 0, \ldots, f_m = 0)$$

over the reals in variables $x_1, \ldots, x_n$, a *Positivstellensatz derivation* of a polynomial inequality $h \geq 0$ is an explicit polynomial expression over $\mathbb{R}$ of the form

$$\sum_{i=1}^{t} q_i \cdot \prod_{j\in A_i} h_i + \sum_{i\in[m']} r_i \cdot f_i = h. \tag{1.7}$$

Where for each $i \in [t]$, $A_i \subseteq [m]$ and q_i is a sum of squares of polynomials. The degree of the proof is the maximum of $\max_i \deg(q_i \cdot \prod_{i \in A_i} h_i$ and $\max_i \deg(r_i \cdot f_i)$

Ordinary **SoS** proofs are Positivstellensatz proofs in which each A_i has size at most 1.

It is not clear whether there are analogous algorithms to those for **SoS** to find low degree Positivstellensatz proofs.

Positivestellensatz calculus proofs

Positivstellensatz calculus proofs are the dynamic generalization of **SA**, **SoS**, and Positivstellensatz proofs. (This is analogous to the way that $\mathbf{PC}_{\mathbb{F}}$ generalizes Nullstellensatz proofs and has the potential to provide substantial savings in degree and proof size.) For simplicity, we assume that each line f represents the statement that $f \geq 0$. Given $h_1 \geq 0, \ldots, h_m \geq 0$ we have the axioms:

$$\text{(Trivial axiom)} \ \frac{}{1}$$

$$\text{(Input axiom)} \ \frac{}{h_i} \ \text{for } i \in [m]$$

We give a non-minimal set of inference rules for simplicity. We will always assume that we are operating mod the Boolean ideal I so we have:

$$\text{(Multilinearization)} \ \frac{f}{\text{multilinear}(f)}$$

$$\text{(Linear Combination)} \ \frac{f \qquad g}{a \cdot f + b \cdot g} \ \text{for } a, b \geq 0$$

$$\text{(Literal)} \ \frac{f}{y \cdot f} \ \text{for } y \in \{x_1, \ldots, x_n, (1-x_1), \ldots, (1-x_n)\}$$

$$\text{(Multiply by Square)} \ \frac{f}{p^2 \cdot f} \ \text{for } p \in \mathbb{R}[x_1, \ldots, x_n]$$

With these rules we can directly simulate **SoS**.

$$\text{(Input Multiplication)} \ \frac{f}{f \cdot h_i} \ \text{for } i \in [m]$$

With the Input Multiplication rule we can directly simulate ordinary Positivstellensatz (and not bother including any axiom but the trivial one). Input Multiplication is a special case a more general rule

$$\text{(Multiplication)} \ \frac{f \qquad g}{f \cdot g}$$

It is unclear how the specific rules affect the strength of the proof system. There are no lower bounds knowm for any version that includes the Multiply by Square rule.

Cone Proof Systems

These are a generalization of both the Ideal Proof Systems and the Positivstellensatz Calculus,

1.6 Interesting classes of formulas

We state all of these as unsatisfiable CNF formulas, which generally mean the negation of the principle under consideration.

Induction Principle This is the CNF formula IND_n, whose clauses are:

- x_1
- $\neg x_i \vee x_{i+1}$ for $i \in [n-1]$
- $\neg x_n$

IND_n has a trivial tree resolution refutation of height n and size n that can be found by unit propagation from either x_1 or $\neg x_n$. An alternative tree resolution refutation of height $\log_2 n$ and size n follows using a binary search tree, where one considers the left half if $x_{\lceil n/2 \rceil} = 0$ and the right half if $x_{\lceil n/2 \rceil} = 1$.

The Pigeonhole Principle The basic pigeonhole principle states that if $m > n$ there is no 1-1 mapping from $[m]$ to $[n]$. For $m > n$, the basic form PHP_n^m of the pigeonhole principle has variables x_{ij} for $i \in [m]$ and $j \in [n]$ and clauses

- Pigeon Clauses: $x_{i1} \vee \ldots \vee x_{in}$ for each $i \in [m]$.
- Hole Clauses: $\neg x_{ij} \vee \neg x_{i'j}$ for each $i \neq i' \in [m]$ and each $j \in [n]$.

PHP_n^m asserts the more general statement where the mapping can be an arbitrary relation. If we want the weaker statement that applies only to functions, we have $function\text{-}PHP_n^m$ which adds the clauses for PHP_n^m:

- Function Clauses: $\neg x_{ij} \vee \neg x_{ij'}$ for each $i \in [m]$ and each $j \neq j' \in [n]$.

Alternatively, we can assert $onto\text{-}PHP_n^m$ which only applies the pigeonhole principle to surjective relations by adding the following to PHP_n^m:

- Surjectivity Clauses: $x_{1j} \vee \ldots x_{nj}$ for each $j \in [n]$.

Finally, there is the $bijective\text{-}PHP_n^m$ which adds both the Function and Surjectivity clauses.

We will see that PHP_n^{n+1} is hard for many proof systems. In fact, PHP_n^m can be shown to be hard for resolution even when m is exponentially large in n. However, PHP_n^m is easy to refute in some semi-algebraic proof systems.

Observe that the standard semi-algebraic translation of PHP_n^m is

- Pigeon inequalities: $\sum_{j=1}^{n} x_{ij} \geq 1$ for every $i \in [m]$
- Hole inequalities: $x_{ij} + x_{i'j} \leq 1$ for every $i \neq i' \in [m]$ and each $j \in [n]$.

We now see that there is a **CP**$*$ refutation of PHP_n^m with $O(mn)$ lines using the following proposition.

Proposition 1.36. *For $2 \leq k \leq m$ there is a k-line* **CP**$*$ *derivation from the hole inequalities of $\sum_{i\in[k]} x_{ij} \leq 1$.*

Proof. We show this by induction on k. The case for $k = 2$ is already an axiom. Then we have

(1) $\sum_{i\in[k]} x_{ij} \leq 1$ Inductive hypothesis

(2) $x_{ij} + x_{(k+1)j} \leq 1$ Given for all $i \in [k]$

Adding $k-1$ times (1) to the sum of all k inequalities in (2) yields

(3) $\sum_{i\in[k+1]} k \cdot x_{ij} \leq 2k-1.$

Applying the division rule we obtain

(4) $\sum_{i\in[k+1]} x_{ij} \leq \lfloor (2k-1)/k \rfloor = 1,$

as required. □

To complete the refutation of PHP_n^m, summing up all m pigeon inequalities, we obtain $\sum_{i\in[m],j\in[n]} x_{ij} \geq m$, while summing up the n derived inequalities from Proposition 1.36 we obtain $\sum_{i\in[m],j\in[n]} x_{ij} \leq n$ and so $n \geq m$ and hence $0 \geq m-n$ which is a contradiction since $n < m$.

SoS similarly can refute PHP_n^m in small size and degree.

Proposition 1.37. *There is a degree 3* **SoS** *refutation of PHP_n^m.*

Proof. Observe that for each $j \in [n]$, we can combine the hole axioms using an **SoS** derivation as follows:

$$(1 - \sum_{i\in[m]} x_{ij})^2 + \sum_{i,i'\in[m],\ i\neq i'} x_{ij}^2 \cdot (1 - x_{ij} - x_{i'j}) \equiv_I 1 - \sum_{i\in[m]} x_{ij}$$

Therefore, summing these for all values of j we get a degree 3 derivation of

$$n - \sum_{i\in[m],\ j\in[n]} x_{ij}.$$

On the other hand, adding all the pigeons axioms together, we obtain

$$\sum_{i\in[m],\ j\in[n]} x_{ij} - m.$$

Adding the two together with $m-n-1$ which is non-negative, we derive -1 as required. □

Pebbling formulas Let $G = (V, E)$ be a directed acyclic graph (DAG) with a single sink t and define $pred(v) = \{u \mid (u, v) \in E\}$. We can define the pebbling formula $PEB(G)$ whose clauses are:

- x_v for every source vertex $v \in V$.
- $(\vee_{u \in pred(v)} \neg x_u) \vee x_v$ For all $v \in V$ that is not a source.
- $\neg x_t$.

Observe that the induction principle is the special case of pebbling formulas where G is a single directed path on n nodes.

$PEB(G)$ also has simple tree resolution refutations via unit propagation of length $|E|$. However, in the case that G has small in-degree there is an interesting version is a related *lifted* version $PEB(G)^{\vee 2}$ in which each variable x_v has been replaced by an OR of two variables $x_v^0 \vee x_v^1$, obtaining a formula whose clauses are:

- $x_v^0 \vee x_v^1$ for every source vertex $v \in V$.
- $(\vee_{u \in pred(v)} \neg x_u^{b(u)}) \vee x_v^0 \vee x_v^1$ for all non-source $v \in V$ and all $b : pred(v) \to \{0, 1\}$.
- $\neg x_t^b$ for $b \in \{0, 1\}$.

For each non-source vertex v, there are $2^{|pred(v)|}$ clauses in this formula of the second type.

$PEB(G)^{(\vee 2)}$ still has resolution refutations of length that is linear in its size as follows:

We will iteratively derive $x_v^0 \vee x_v^1$ for each $v \in V$. We already have this for the source vertices. To derive it for a new vertex v, we eliminate the predecessors of vertex v, one at a time. In particular, let $u^* \in pred(v)$ and fix b on $pred(v) - u^*$. We have clauses

$$C_0 = \neg x_{u^*}^0 \vee (\vee_{u \in pred(v) - u^*} \neg x_u^{b(u)}) \vee x_v^0 \vee x_v^1$$
$$C_1 = \neg x_{u^*}^1 \vee (\vee_{u \in pred(v) - u^*} \neg x_u^{b(u)}) \vee x_v^0 \vee x_v^1$$

and we have $C_{u^*} = x_{u^*}^0 \vee x_{u^*}^1$ by the inductive hypothesis. We first resolve C_{u^*} and C_0 to get

$$D = x_{u^*}^1 \vee (\vee_{u \in pred(v) - u^*} \neg x_u^{b(u)}) \vee x_v^0 \vee x_v^1$$

and then resolving D and C_1 yields

$$(\vee_{u \in pred(v) - u^*} \neg x_u^{b(u)}) \vee x_v^0 \vee x_v^1.$$

Repeating this for all assignments b on $pred(v) - u^*$ and then repeatedly choosing elements of $pred(v)$ that remain, one-by-one, yields the desired clause $x_v^0 \vee x_v^1$ in $O(2^{|pred(v)|})$ steps, which is linear in the number of clauses associated with vertex v.

It will turn out that instances of these formulas on certain graphs hards graphs G of bounded in-degree have useful properties. The following game, which was originally defined to understand the number of registers required to evaluate circuits, gives the key property.

Definition 1.38. The (black) *pebbling game* is a single-player game played on the nodes of a directed acyclic graph G with a single sink t according to the following rules:

- The graph G is initially empty.
- If all predecessors of a node v have pebbles then a pebble may be placed on v.
- A pebble may be removed from a node at any time.
- The game ends when a pebble has been placed on t.

The (black) *pebbling number*, $\text{peb}(G)$ of a graph G is the minimum over all strategies of the maximum number of pebbles that need to be on the vertices of G at any one time.

Binary trees on n nodes have pebbling number $O(\log n)$. Other graphs can have much larger pebbling number. For example, the n-node pyramid graph which is triangle consisting of a corner of the square grid with the origin at the root has pebbling number $\Omega(\sqrt{n})$. In fact, there are graphs of in-degree 2 that have pebbling number $\Omega(n/\log n)$, which is optimal.

Ordering principles There are several such principles, all of which use variables x_{uv} for $u \neq v \in V$ for some set V to indicate that $u \prec v$ for some total order $\prec$ on V. They all contain the following axioms:

- Totality: $x_{uv} \vee x_{vu}$ for all $u \neq v \in V$.
- Anti-symmetry: $\neg x_{uv} \vee \neg x_{vu}$ for all $u \neq v \in V$.
- Transitivity: $\neg x_{uv} \vee \neg x_{vw} \vee x_{uw}$ for all distinct $u, v, w \in V$.

The simplest of these formulas, GT_n, has $V = [n]$ and adds axioms enforcing that the total order $\prec$ does not have a minimal element:

- Non-minimality: $\bigvee_{u \neq v} x_{uv}$ for all $v \in V$.

More generally, if $G = (V, E)$ is an undirected graph, the *graph ordering principle* on G, $GOP(G)$, replaces the non-minimality axioms with a local version stating that no vertex is even a *local* minimum of $\prec$.

- Local non-minimality: $\bigvee_{u:(u,v)\in E} x_{uv}$ for each $v \in V$.

Note that $GT_n = GOP(K_n)$ where K_n is the complete graph.

Proposition 1.39. *GT_n has a polynomial size ordered resolution refutation.*

Proof. The general idea is to derive the non-minimality axioms of GT_{n-1} from those of GT_n by resolving out all variables that touch vertex n, one after another. The details are left as an exercise. □

Tseitin formulas These formulas are based on the handshaking principle for undirected graphs. Tseitin originally introduced these formulas in his 1968 paper in which he first defined regular resolution. He used these formulas over grid graphs as examples for which he proved the first super-polynomial lower bounds for regular resolution.

There is a Tseitin formula $TS(G,\ell)$ for each undirected graph $G=(V,E)$ and each labeling function $\ell : V \to \{0,1\}$ with total parity being *odd*, that is $\sum_{v\in V} \ell(v) \equiv 1 \pmod 2$. It has a variable x_e for each edge $e \in E$ and has:

- clauses representing the constraint that $\sum_{e\ni v} x_e \equiv \ell(v) \pmod 2$ for each $v \in V$.

  (If vertex v has degree d, there will be 2^{d-1} clauses representing the constraint at v.)

By the handshaking principle, $TS(G,\ell)$ would require that

$$2|E| = \sum_{v\in V}\sum_{e\ni v} x_e \equiv \sum_{v\in V} \ell(v) \equiv 1 \pmod 2$$

and hence is unsatisfiable.

Given two odd labelings ℓ and ℓ' of V, it is quite easy to derive $TS(G,\ell)$ from $TS(G,\ell')$, so the complexity of the problem generally depends only on the graph G

The Tseitin formulas are of particular interest for constant-degree graphs, for which the formulas have $O(m)$ clauses. We will see that they are particularly difficult to refute over constant-degree *expander graphs*.

Modular counting formulas These are defined for any integer $k \geq 2$ and integer n not divisible by k. They express the property that $[n]$ cannot be partitioned into sets of size k. The modular counting formula $COUNT_k^n$ has variables x_e for each $e \subset [n]$ with $|e| = k$; for two such sets $e, f \subset [n]$, we write $e \perp f$ iff $0 < |e \cap f| < k$. The clauses of $COUNT_k^n$ are:

- $\bigvee_{e\ni v} x_e$ for each $v \in [n]$.
- $\neg x_e \vee \neg x_f$ for each $e, f \subset [n]$ with $e \perp f$.

The formula $COUNT_2^{2n+1}$ is also known as the *Parity Principle* and essentially says that the graph K_{2n+1} does not have a perfect matching. Observe that the unsatisfiability of $bijective\text{-}PHP_n^{n+1}$ follows from that of the Parity Principle.

Proposition 1.40. *For p a prime and n not divisible by p, there are polynomial-size Nullstellensatz refutations of $COUNT_p^n$ in $\mathbf{PC}_{\mathbb{F}_p}$.*

Random k-CNF formulas Let $\mathscr{F}_n^{k,m}$ be distribution of random k-CNF formulas with m clauses chosen uniformly randomly and independently from the set of $2^k\binom{n}{k}$ possible clauses on k distinct variables.

Unlike the other families of CNF formulas we consider, random k-CNF formulas are not always unsatisfiable. However, if there are sufficiently many clauses, then they are almost surely unsatisfiable:

Theorem 1.41. *Let $k \geq 2$ be an integer and $\Delta > 2^k \ln 2$. For $F \sim \mathscr{F}_n^{k,m}$ where $m \geq \Delta n$, F is unsatisfiable with probability $1 - o_n(1)$.*

Proof. Let $\varepsilon = \Delta - 2^k \ln 2$. Fix a single truth assignment α to the variables of $F \sim \mathscr{F}_n^{k,m}$. A uniformly random k-clause is satisfied by α with probability precisely $1 - 1/2^k$. Since k-clauses of $F \sim \mathscr{F}_n^{k,m}$ are chosen uniformly and independently, the probability that α satisfies F is precisely $(1-1/2^k)^m$. Let $\#(F)$ be the number of satisfying assignments for F. Therefore

$$\begin{aligned}
\Pr_{F\sim\mathscr{F}_n^{k,m}}[F \text{ is satisfiable }] &\leq \mathop{\mathbb{E}}_{F\sim\mathscr{F}_n^{k,m}}[\#(F)] \\
&= \sum_{\alpha\in\{0,1\}^n} \Pr_{F\sim\mathscr{F}_n^{k,m}}[\alpha \text{ satisfies } F] \\
&= 2^n \cdot (1-1/2^k)^m \leq 2^n\, e^{-m/2^k} \\
&\leq 2^n \cdot e^{-n\ln 2 - \varepsilon\, n/2^k} = e^{-\varepsilon n/2^k},
\end{aligned}$$

which is $o_n(1)$ since k is fixed. □

2
The Complexity of Resolution

2.1 An exponential lower bound for the pigeonhole principle

The first exponential lower bound for general resolution was proven for the pigeonhole principle PHP_n^{n+1} by Haken in 1984, who introduced many of the concepts important for resolution lower bounds. Here we present a simpler proof that retains many of these ideas and gets a slightly sharper bound.

Theorem 2.1. *For $n \geq 2$, any resolution refutation of PHP_{n-1}^n has size at least $2^{n/20}$.*

Proof. Following Haken, a truth assignment to the variables x_{ij} of PHP_{n-1}^n is called *critical* if it defines a one-to-one, onto map from $n-1$ pigeons to $n-1$ holes, with the remaining pigeon not mapped to any hole. A critical assignment where i is the pigeon left out is called *i-critical*. In what follows we will only be interested in critical truth assignments.

Let C be a clause. The monotone clause $M(C)$ associated to C is obtained by replacing each occurrence of a negative literal $\neg x_{ik}$ by the set of literals $\{x_{i'k} \mid i' \neq i\}$. It is easy to check that C and $M(C)$ are satisfied by precisely the same set of critical assignments.

We will be interested in restrictions corresponding to partial matchings that one obtains by repeatedly choosing an (i, j) pair, and setting $x_{ij} = 1$, $x_{ij'} = 0$ for $j' \neq j$, and $x_{i'j} = 0$ for $i' \neq i$. Observe that if one begins with a resolution refutation of PHP_{n-1}^n and one chooses such a partial matching restriction that sets t variables to 1 then the result is resolution refutation of PHP_{n-t-1}^{n-t}. Furthermore, the restriction applied to the monotone conversion of each clause results in the same clause as doing the monotone conversion of the clause first and then applying the restriction. (The transformation to monotone clauses, is not essential, but makes the argument slightly cleaner.)

So let $C_1, ..., C_S$ be a resolution refutation of PHP_{n-1}^n and $M_1 = M(C_1), \ldots, M_S = M(C_S)$ be its monotone conversion. Say that a converted clause M_t is *large* if it has at least $n^2/10$ (positive) literals, i.e. at least 1/10-th of all the

variables. To show that $S \geq 2^{n/20}$, we will show that the number L of large clauses is at least $2^{n/20}$. Assume for contradiction that $L < 2^{n/20}$. Let d_{ij} denote the number of large clauses containing x_{ij}. By averaging, there is an (i,j) with $d_{ij} \geq L/10$. Choose such an (i,j) pair, and apply the restriction $x_{ij} = 1$, $x_{ij'} = 0$ for $j' \neq j$, and $x_{i'j} = 0$ for $i' \neq i$. Applying this restriction we obtain a monotone conversion of a refutation of PHP^{n-1}_{n-2} with at most $9L/10$ large clauses. Applying this argument iteratively $\log_{10/9} L$ times, we are guaranteed to have knocked out all large clauses. Thus, we are left with a refutation of $PHP^{n'}_{n'-1}$, where

$$n' \geq n - \log_{10/9} L = (1 - (\log_{10/9} 2)/20)n > 0.671n,$$

and where no clause in the refutation is large. But this contradicts the following lemma (originally due to Haken [1]) which states that such a refutation must have a clause whose monotone conversion has size at least $2(n')^2/9 > n^2/10$.

Lemma 2.2. *Any resolution refutation of PHP^n_{n-1} must contain a clause C such that $M(C)$ has at least $2n^2/9$ literals.*

Proof. Given a clause C, let

$$badpigeons(C) = \{i \mid \text{there is some } i\text{-critical assignment } \alpha \text{ falsifying } C\}.$$

Define the complexity $comp(C) = |badpigeons(C)|$.

Let P be a resolution refutation of PHP^n_{n-1}, and consider the complexity of the clauses that appear in P. The complexity of each initial clause is at most 1 and the complexity of the final false clause is n.

Note that if we use the resolution rule to derive a clause C from two previous clauses C' and C'', we have that $comp(C) \leq comp(C') + comp(C'')$, since any assignment falsifying C must also falsify at least one of C' or C''. If C is the first clause in the proof with $comp(C) > n/3$, we must have $n/3 < comp(C) \leq 2n/3$. We will show that $M(C)$ contains a large number of variables.

For $comp(C) = t$ will now show that $M(C)$ has at least $(n-t)t \geq 2n^2/9$ distinct literals mentioned. Fix some $i \in badpigeons(C)$, and let α be an i-critical truth assignment falsifying C. For each $j \notin badpigeons(C)$, consider the j-critical assignment, α', obtained from α by replacing i by j, that is by mapping i to the place that j was mapped to in α. This assignment satisfies C, and differs from α only in one place: if α mapped j to k, then α' maps i to k. Since C and $M(C)$ agree on all critical assignments and $M(C)$ is monotone, it must contain the variable x_{ik}.

Running over all $n-t$ j's not in $badpigeons(C)$ (using the same α), it follows that $M(C)$ must contain at least $n-t$ distinct variables x_{ik}, $k \leq n$. Repeating the argument for all $i \in badpigeons(C)$ shows that C contains at least $(n-t)t$ positive literals. □

□

2.2 Resolution width and proof size

Definition 2.3. For a CNF formula F, we write $\text{Res}(F)$ for the minimum number of clauses in any Resolution refutation of F. If F is satisfiable then $\text{Res}(F) = \infty$.

We define $\text{Res}_{tree}(F)$, $\text{Res}_{order}(F)$, and $\text{Res}_{reg}(F)$ analogously for the the number of clauses required for the restricted cases of tree resolution, ordered (Davis-Putnam) resolution, and regular resolution proofs respectively.

Definition 2.4. For a CNF formula F write $w(F) = \max_{C \in F} |C|$ and for a resolution proof P, define the *width* of P, $w(P) = \max_{C \in P} |C|$.

For a CNF formula F define $\text{width}(F)$ to be the minimum width $w(P)$ over all resolution refutations P of F (and ∞ if no such refutation exists).

Remark 2.5. Observe that any CNF formula F has a tree-like resolution refutation of width $\text{width}(F)$, since, by repeating derivations, any resolution refutation can be expanded to a tree-like one using the same set of clauses.

Definition 2.6. A *restriction* on set of variables X is a partial assignment of Boolean values $\rho : X \to \{0, 1, *\}$, where $\rho(x) = 0$ or $\rho(x) = 1$ indicates that variable $x \in X$ has been set to 0 or 1, respectively, and $\rho(x) = *$ indicates that variable x is unaffected.

Given a Boolean formula F and a restriction ρ defined on the variables of F, we write F^ρ for the formula in which each variable x of F assigned by ρ is substituted by $\rho(x)$; we write $F|_\rho$ for the formula in which we make substitutions as in F^ρ and then apply all immediate simplifications based on the substituted values. In particular, if F is a CNF formula, in computing $F|_\rho$, every clause satisfied by ρ is removed and every falsified literal is removed from the remaining clauses of F. Similarly, if P is a resolution proof, in $P|_\rho$, every satisfied clause is removed and any remaining clause is shortened.

We sometimes describe a restriction ρ as an explicit sequence of pairs of the form $x \leftarrow \rho(x)$ or $\neg x \leftarrow 1 - \rho(x)$ for all $x \in X$ for which $\rho(x) \neq *$. In that case, we omit the vertical bar in the notation where we apply the restriction, so that, for example, if ρ is $x \leftarrow 1, y \leftarrow 0$ we write $F_{x \leftarrow 1, y \leftarrow 0}$ in place of $F|_\rho$.

Proposition 2.7. *(a) Let P be a resolution derivation of a clause C from CNF formula F. For any restriction ρ on the variables of F, $P|_\rho$ is a resolution derivation of $C|_\rho$ from $F|_\rho$.*

(b) For literal z, if $\text{width}(F_{z \leftarrow 1}) \leq w$ then either $\text{width}(F) \leq w$ or there is a resolution derivation of $\neg z$ from F of width at most $w + 1$.

Proof. Part (a) is immediate. For part (b), let P' be a refutation of $F_{z \leftarrow 1}$ of width at most w. For the new derivation, replace each input clause of $F_{z \leftarrow 1}$ by the corresponding clause of F and retain the same sequence of resolution steps (which is possible since none of the resolution steps involve the literal

z) to yield a proof P. The replacement of input clauses of $F_{z\leftarrow 1}$ by those of F may add $\neg z$ to some input clauses. It is immediate, inductively, that every clause of proof P' either stays the same or has $\neg z$ added to it in P. If the output clause is still the empty clause $\perp$, then all the clauses of $F_{z\leftarrow 1}$ leading to the output clause of P' were in the original formula F, so P' is the required refutation. Otherwise, the clause width of P is at most $w+1$ and the output clause is $\neg z$. □

Theorem 2.8. *Let F be a CNF formula in n variables.*

(a) If $\mathrm{Res}_{tree}(F) \le S$ *then* $\mathrm{width}(F) \le \log_2 S + w(F)$.

(b) If $\mathrm{Res}(F) \le S$ *then* $\mathrm{width}(F) \le \max(2\sqrt{2n\ln S}, \sqrt{2n\ln S} + w(F))$.

Proof. We first prove part (a) by induction on S: For the base case, observe that it holds for $S = 1$ since in that case the proof width is 0.

Let P be a tree-like refutation of F of size at most S and assume that the final resolution step (producing the empty clause $\perp$) resolves clauses x and $\neg x$. Since P is tree-like, the derivations of x and $\neg x$ do not share any instances of derived clauses and so P consists of two disjoint derivations, P_x that infers x from F and $P_{\neg x}$ that infers $\neg x$ from F. By definition, $\#(P_x) + \#(P_{\neg x}) \le S - 1$. Hence, both P_x and $P_{\neg x}$ have at most $S-1$ clauses and at least one has at most $S/2$ clauses. Let z be a literal (equal to x or $\neg x$) for which $\#(P_{\neg z}) \le S/2$.

Applying the restriction $z \leftarrow 1$ to $P_{\neg z}$, by Proposition 2.7(a) we obtain a tree-like length $\le S/2$ derivation of $\perp$ from $F_{z\leftarrow 1}$. Hence we can apply our inductive hypotheses to derive a refutation P_1' of $F_{z\leftarrow 1}$ with $w(P_1') \le \log_2 S + w(F) - 1$. Applying Proposition 2.7(b) to P_1' and z, we obtain a derivation P_1'' of $\neg z$ from F with $w(P_1'') \le \log_2 S + w(F)$.

Applying the restriction $z \leftarrow 0$ to P_z, we similarly obtain a tree-like length $\le S-1$ derivation of $\perp$ from $F_{z\leftarrow 0}$ and hence by the inductive hypothesis we can efficiently compute a refutation P_0' of $F_{z\leftarrow 0}$ with $w(P_0') \le \log_2 S + w(F)$.

The refutation of F of width at most $\log_2 S + w(F)$ is constructed as follows: (1) use the steps of P_1'' to derive $\neg z$ then (2) resolve $\neg z$ with clauses of F containing z to produce clauses of $F_{z\leftarrow 0}$, and finally (3) use the steps of P_0' to derive $\perp$ from $F_{z\leftarrow 0}$.

We now prove part (b):
Set $W = \lceil \sqrt{2n\ln S} \rceil$. We say that a clause C is *wide* iff $w(C) \ge W$. We prove the following claim by induction on n and k:

CLAIM: If $(1-W/2n)^k\, S < 1$ then any CNF formula F in n variables having a resolution refutation with $\le S$ wide clauses has $\mathrm{width}(F) \le \max(W, w(F)) + k$.

Before proving the claim, we observe that the case $k = W$ is sufficient to prove part (b), since $(1 - W/2n)^W < e^{-W^2/2n} \le 1/S$ by the choice of W.

The case $k = 0$ is trivial, since a refutation with no wide clauses has width at most W.

For the general case, let P be a resolution refutation of F with n variable and $\leq S$ clauses of width $\geq W$ and suppose that $(1-W/2n)^k S \leq 1$. Choose the literal z appearing in the most wide clauses of P. Since there are $\leq 2n$ possible literals and $\geq W$ distinct literals per wide clause, z appears in $\geq WS/2n$ wide clauses.

Consider the restrictions $z \leftarrow 1$ and $z \leftarrow 0$: Then $P_{z\leftarrow 1}$ is a resolution refutation of $F_{z\leftarrow 1}$ and every clause of P containing z is satisfied and hence removed (and others are only shortened), so $P_{z\leftarrow 1}$ has $S' \leq S - WS/2n = (1-W/2n)\,S$ wide clauses. Therefore $(1-W/2n)^{k-1} S' \leq (1-W/2n)^k S \leq 1$. It follows by our inductive hypothesis with $k' = k-1$, that

$$\text{width}(F_{z\leftarrow 1}) \leq \max(W, w(F_{z\leftarrow 1})) + k - 1 \leq \max(W, w(F)) + k - 1.$$

By Proposition 2.7(b), either $\text{width}(F) \leq \max(W, w(F)) + k$, and we are done, or there is a derivation P' of $\neg z$ from F of width $\leq \max(W, w(F)) + k$.

Now, $P_{z\leftarrow 0}$ is a refutation of $F_{z\leftarrow 0}$. By the inductive hypothesis applied to $F_{z\leftarrow 0}$, which has $n' = n-1$ variables, there is a refutation P'' of $F_{z\leftarrow 0}$ of width at most $\max(W, w(F)) + k$. We next resolve $\neg z$ with clauses of F containing z to produce every clause of $F_{z\leftarrow 0}$ used in P''. This part requires width at most $w(F)$.

Putting the parts together we obtain a single refutation of F of width at most $\max(W, w(F)) + k$ as required for the induction step. □

Corollary 2.9. *Let F be a CNF formula in n variables.*

- *If there is a resolution refutation of F or size at most S then there is an algorithm running in time $n^{O(\sqrt{n \log S} + w(F))}$ that will find a resolution refutation of F.*
- *If there is a tree resolution refutation of F or size at most S then there is an algorithm running in time $S^{O(\log n)} n^{O(w(F))}$ that will find a resolution refutation of F.*

Proof. For each bound, apply a width-increasing search on proofs up to the width bound W^* given in Theorem 2.8. That is, it applies all possible resolution inferences that yield clauses of width w beginning with $w = w(F)$ and increasing until a refutation is found. The number of distinct clauses up to this width bound W^* is less than $\sum_{w=0}^{W^*} \binom{2n}{w}$ and the running time is polynomial in this number of potential clauses. Plugging in the different values of W^* and using standard binomial bounds for the two cases yields the claimed running times. □

We restate Theorem 2.8 in the following convenient form:

Theorem 2.10. *For any CNF formula F in n variables,*

(a) $\mathrm{Res}_{tree}(F) \geq 2^{\mathrm{width}(F)-w(F)}$, *and*

(b) $\mathrm{Res}(F) \geq \exp[\min((\mathrm{width}(F)-w(F))^2/2n, \mathrm{width}(F)^2/8n)]$.

This implies that sufficiently strong resolution width lower bounds suffice for proving resolution size lower bounds. It is known that the width-size relationship in Theorems 2.8 and 2.10 cannot be improved beyond a logarithmic factor in width or a polynomial factor in size.

In the following sections we show that certain *expansion* properties of the clauses of F imply good width lower bounds.

2.3 Boundary expansion and bipartite expansion

Definition 2.11. For a bipartite graph $G = (L, R, E)$ and a set $S \subseteq L$, the *boundary* of S, denoted ∂S, is the set of all $v \in R$ that have exactly one neighbor $u \in S$.

Graph $G = (L, R, E)$ is an (r, c)-*boundary expander* iff for every $S \subset L$ with $|S| \leq r$, the boundary ∂S satisfies $|\partial S| \geq c|S|$.

Definition 2.12. Any CNF formula F corresponds to a bipartite graph $G_F = (L, R, E)$ where L is the set of clauses of F, R is the set of variables of F, and $(C, x) \in E$ iff variable x appears in clause C.

Given a set of clauses S, the boundary of S, ∂S, in G_F is a set of variables, but since each occurs with a unique sign in the clauses of S, we can also interpret ∂S as a set of literals when it is convenient.

Lemma 2.13. *If F is a CNF formula, D is a clause and*

- *there is a resolution derivation of C^* from F for which S is the set of clauses of F that have a path to C^*, or*
- *S is a minimal subset of the clauses of F whose conjunction implies clause C^**

then $|C^*| \geq |\partial S|$.

Proof. For the first case, observe that the literals in the boundary variables of S pass through to C^* without cancellation. For the second case, let x be a variable in ∂S and C be the unique clause in S containing x in the form of literal ℓ. The minimality of S implies that there is some assignment that satisfies all clauses in $S - \{C\}$ but makes C^* false. We can also make all of S true by flipping ℓ to true, without changing the assignment on the remaining clauses, which in turn must make C^* true. Hence ℓ is in C^*, so $|C^*| \geq |\partial S|$. □

Boundary expansion plays a role in a wide variety of lower bound arguments in proof complexity. In particular, it suffices to prove width lower bounds for resolution proofs and hence lower bounds on resolution and tree resolution proof size using Theorem 2.10.

Theorem 2.14. *Any CNF formula F for which G_F is an (r,c)-boundary expander requires resolution refutation width $> cr/2$.*

Proof. Without loss of generality we may assume that $c > 0$. Define the *complexity* of a clause C in the refutation of F to be the size of the subset of clauses S of F that have a path to C in the proof. In particular, since G_F is an (r,c)-boundary expander, any set S of clauses of size at most r has $|\partial S| > c|S| > 0$ by Lemma 2.13 and hence $C \neq \bot$. Therefore, the empty clause at the root of the proof must have complexity $> r$. The input clauses have complexity 1. By the soundness of the resolution rule, complexity is sub-additive; that is, if C is derived from A and B, then the complexity of C is at most the sum of the complexities of A and B.

We therefore follow the proof back from the root, always taking the branch of larger complexity, which will be at least $1/2$ of the previous complexity by sub-additivity. Eventually this must pass through a clause C^* with paths from a minimal subset S of clauses of F with $r/2 < |S| \leq r$ that yields C^*. Since G_F is an (r,c)-bipartite expander, $|C^*| \geq |\partial S| \geq c|S| > cr/2$ as required. □

In general, we can derive boundary expansion as a consequence of sufficiently large bipartite expansion (relative to clause size).

Definition 2.15. A bipartite graph $G = (L,R,E)$ is an (r,c)-*bipartite expander* iff every set $S \subseteq L$ with $|S| \leq r$ has $|N(S)| \geq (1+c)|S|$, where $N(S)$ is the set of neighbors of elements of S in G.

Proposition 2.16. *If $G = (L,R,E)$ has left degree at most k and is an (r,c)-bipartite expander, then it is an (r,c')-boundary expander for $c' = 2(c+1)-k$.*

Proof. Let $S \subset L$ with $|S| \leq r$. Every element of the neighborhood $N(S)$ of S that is not in ∂S is incident to at least 2 of the at most $k|S|$ edges that are incident to S. Therefore

$$k|S| \geq 2|N(S) - \partial S| + |\partial S| = 2|N(S)| - |\partial S|.$$

Since G is an (r,c)-bipartite expander, $|N(S)| \geq 1 + c = (k + c')|S|/2$ and plugging this in yields $|\partial S| \geq c'|S|$. □

Corollary 2.17. *In particular, if F is k-CNF formula such that G_F is an (r,c)-bipartite expander for $c \geq k/2-1+\delta$ then G_F is an $(r,2\delta)$-boundary expander and hence* width$(F) > r\delta$*,*

Proof. This follows by plugging in $c = k/2-1+\delta$ into Proposition 2.16 and then applying Theorem 2.14. □

For resolution, it turns out that boundary expansion is a more stringent requirement than necessary and bipartite expansion alone suffices. Namely, rather than requiring bipartite expansion constant larger than $k/2-1$, it suffices to merely have positive bipartite expansion.

Theorem 2.18. *Any CNF formula F for which G_F is an (r,c)-bipartite expander requires resolution refutation width $> cr/2$.*

This theorem follows almost exactly the same pattern as the result for boundary expanders but uses stronger properties of the size of derived clauses that follow from the use of Hall's Theorem stated below.

Proposition 2.19 (Hall's Theorem)**.** *In a bipartite graph $G = (L,R,E)$, a subset $S \subseteq L$ can be matched in G if and only if for every subset $T \subseteq S$, $|N(T)| \geq |T|$.*

Proof Sketch. Clearly, if the condition is violated, there is no way to match all the elements of T. The other direction follows from the fact that for any proper subset $T' \subset S$ and any $v \in S \setminus T'$, any matching on T' can be converted to a matching of $T = T' \cup \{v\}$ (via an augmenting path) since $|N(T)| \geq |T| > |T'|$. □

Hall's Theorem yields an important consequence for the sizes of minimally unsatisfiable sets of clause.

Lemma 2.20. *For any minimally unsatisfiable set of clauses S, $|Vars(S)| < |S|$*

Proof. Consider the bipartite graph G_S for the CNF formula on S. If G_S contains a matching on all clauses of S then the assignment that sets each variable according to the clause it is matched to is a satisfying assignment. Since S is unsatisfiable, there is no such matching on S. Therefore, by Hall's Theorem, there is some subset $T \subseteq S$ such that $|N(T)| < |T|$ in G_S, that is, $|Vars(T)| < |T|$. Choose the largest such subset T.

Suppose that $T \neq S$. Then by the minimality of S there is some assignment α to $N(T)$ that satisfies T. By the maximality of T, for every nonempty $V \subseteq S - T$, we have $|N(T \cup V)| \geq |T \cup V|$ and hence $|N(V) - N(T)| > |V|$. (We don't actually need strict inequality here.) By Hall's Theorem, this means that there is a matching of the clauses in $S - T$ to the variables in $N(S) - N(T)$. If we combine α with the partial assignment β to the variables in $N(S) - N(T)$ that agrees with the clause it is matched to, we obtain a satisfying assignment to S, which is a contradiction. □

The following somewhat technical, but easy, consequence is the key to the sharper lower bounds based on bipartite expansion.

Lemma 2.21. *If F is a CNF formula, D is a clause and S is a minimal subset of the clauses of F whose conjunction implies D, then $|D| > |Vars(S)| - |S|$.*

Proof. Let ρ be the unique partial truth assignment of size $|D|$ that falsifies D. Consider the set of clauses $S|_\rho = \{C|_\rho \mid C \in S\}$. This must be contradictory since implication is preserved under the application of any restriction. Moreover, for any proper subset $S' \subset S$ if $S'|_\rho$ is contradictory, then the

clauses in S' would imply D, contradicting the minimality of S. Therefore, $S|_\rho$ is a minimally unsatisfiable set of clauses and hence, by Lemma 2.20,

$$|Vars(S)| - |D| = |Vars(S|_\rho)| < |S|_\rho| \leq |S|. \qquad \square$$

We are now ready to prove Theorem 2.18.

Proof of Theorem 2.18. Without loss of generality we may assume that $c > 0$. Define the *complexity* of a clause D in the refutation of F as the minimal size of a subset of clauses S of F that implies D. In particular, since G_F is an (r, c)-bipartite expander, any set S of clauses of size at most r has $|Vars(S)| > |S|$ and hence is satisfiable by Lemma 2.20. Therefore, the empty clause at the root of the proof must have complexity $> r$. The input clauses have complexity 1. By the soundness of the resolution rule, complexity is sub-additive; that is, if D is derived from A and B, then the complexity of D is at most the sum of the complexities of A and B.

We therefore follow the proof back from the root, always taking the branch of larger complexity, which will be at least $1/2$ of the previous complexity by sub-additivity. Eventually this must pass through a clause D having a minimal subset S of clauses of size between $r/2$ and r that implies D. Since G_F is an (r, c)-bipartite expander, $|Vars(S)| \geq (1 + c)|S|$ and by Lemma 2.21, we get $|D| > |Vars(S)| - |S| \geq c|S| \geq cr2$ as required. $\square$

2.4 Random k-CNF formulas: Expansion and resolution lower bounds

Let $\mathscr{F}_n^{k,m}$ be distribution of random k-CNF formulas with m clauses chosen uniformly randomly and independently from the set of $2^k\binom{n}{k}$ possible clauses on k distinct variables.

Lemma 2.22. *Let $\delta > 0$. If $0 < (1 + \delta)c < k - 2$, there is a constant $C_{c,\delta} > 0$ such that if $\Delta = \Delta(n) \leq n^{k-2-(1+\delta)c}$ then for $F \sim \mathscr{F}_n^{k,\Delta n}$, G_F is an (r, c)-bipartite expander with probability $1 - o_n(1)$ for $r = C_{c,\delta} \cdot n/\Delta^{1/(k-2-c)}$.*

Proof. Fix a set of clauses S of size s, let $q = (1 + c)s$, and fix a set T of variables of size q. For a single $C \in S$, Let $p = \binom{q}{k}/\binom{n}{k} \leq q^k/n^k$ be the probability that all variables of C are in T. Now

$$\begin{aligned}
\Pr[|N(S)| \leq q] &\leq \binom{n}{q} p^s \\
&\leq \left(\frac{ne}{q}\right)^q \left(\frac{q}{n}\right)^{ks} \\
&= e^q \cdot \left(\frac{q}{n}\right)^{ks-q} \\
&= e^{(1+c)s} \cdot \left(\frac{(1+c)s}{n}\right)^{(k-1-c)s} \\
&= a^s \cdot \left(\frac{s}{n}\right)^{(k-1-c)s}
\end{aligned}$$

for $a = e^{(1+c)}(1+c)^{k-1-c}$. Therefore, the probability that G_F is not an (r,c)-bipartite expander is at most

$$\begin{aligned}\sum_{s=1}^{r}\binom{\Delta n}{s} a^s \cdot \left(\frac{s}{n}\right)^{(k-1-c)s} &\le \sum_{s=1}^{r}\left(\frac{a\cdot e\cdot \Delta n}{s}\right)^s \cdot \left(\frac{s}{n}\right)^{(k-1-c)s} \\ &= \sum_{s=1}^{r}\left[a\cdot e\cdot \Delta\cdot\left(\frac{s}{n}\right)^{k-2-c}\right]^s \\ &= \sum_{s=1}^{r}\left[b\cdot \Delta\cdot\left(\frac{s}{n}\right)^{k-2-c}\right]^s\end{aligned}$$

for $b = a\cdot e$ which depends only on c and k.

To bound this quantity we split the sum into two cases depending on whether s is small or large. More precisely, we set a threshold t that is a small growing function of n and show that $s > t$, we can bound each term by $1/2^s$ so the total of the large terms is $O_n(1/2^t)$ which is $o_n(1)$ and for $s \le t$ we show that it is $o_n(1/t)$ and hence the total is $o_n(1)$.
Set $C = 1/(2b)^{1/(k-2-c)}$. We first bound every term $s \le r$ by $1/2^s$:
Then

$$\begin{aligned}b\cdot\Delta\cdot\left(\frac{s}{n}\right)^{k-2-c} &\le b\cdot\Delta\cdot\left(\frac{r}{n}\right)^{k-2-c} \\ &= b\cdot\Delta\cdot\left(\frac{C}{\Delta^{1/(k-2-c)}}\right)^{k-2-c} \\ &= b\cdot C^{k-2-c} \\ &= b\cdot 1/(2b) = 1/2,\end{aligned}$$

which implies that the term for s is at most $1/2^s$.

Now, we have a lot of freedom to choose t provided that it is $\omega_n(1)$ and not too large. For any such t, the sum of terms over all s such that $t < s \le r$ is at most $\sum_{s\ge t} 1/2^s = 1/2^{t-1}$ which is $o_n(1)$. For example, we can chose $t = \log_2 n$ or $t = n^{\delta c/[2(k-1-c)]}$. We now show that we can get a stronger bound on the total contribution for $s \le t$. Observe that in this case,

$$\begin{aligned}b\cdot\Delta\cdot\left(\frac{s}{n}\right)^{k-2-c} &\le b\cdot n^{k-2-(1+\delta)c}\cdot\left(\frac{t}{n}\right)^{k-2-c} \\ &\le b\cdot t^{k-2-c}\cdot n^{-\delta c}.\end{aligned}$$

There are at most t such terms (each raised to a power $s \ge 1$) so the total contribution of such terms is at most $b\cdot t^{k-1-c}\cdot n^{-\delta c} \le bn^{-\delta c/2}$, which is $o_n(1)$ and hence the sum of terms for all $s \le r$ is $o_n(1)$ as required.

It remains to show that C can be taken to depend on only on c and δ and not on k. Set $C_{c,\delta} = 2^{-1/(\delta c)}\cdot e^{-(2+c)/(\delta c)}\cdot(1+c)^{-(1+1/(\delta c))} > 0$. It suffices to show that $C_{c,\delta} \le C = 1/(2b)^{1/(k-2-c)}$ or equivalently that $C_{c,\delta}^{k-2-c} \le 1/(2b)$.

Since $k-2>(1+\delta)c$ we have $k-2-c \geq \delta c$ and so

$$\begin{aligned} C_{c,\delta}^{k-2-c} &= [2^{-1/(\delta c)} \cdot e^{-(2+c)/(\delta c)} \cdot (1+c)^{-(1+1/(\delta c))}]^{k-2-c} \\ &\leq 2^{-1} \cdot e^{-(2+c)} \cdot (1+c)^{-(k-2-c)} \cdot (1+c)^{-1} \qquad \text{using } k-2-c > \delta c \\ &= 1/(2 \cdot e \cdot e^{1+c} \cdot (1+c)^{k-1-c}) \\ &= 1/(2 \cdot e \cdot a) = 1/(2b) \end{aligned}$$

as required. □

Putting the pieces together to obtain exponential lower bounds for resolution refutations for random k-CNF formulas.

Theorem 2.23. *For $k \geq 3$ and every $\varepsilon' > \varepsilon > 0$, there is a constant $C > 0$ such that if $F \sim \mathscr{F}_n^{k,\Delta n}$ for $\Delta = \Delta(n) \leq n^{k-2-\varepsilon'}$ then with probability $1-o_n(1)$,*

- $\mathrm{Res}(F) > 2^{Cn/\Delta^{2/(k-2-\varepsilon)}}$,
- $\mathrm{Res}_{tree}(F) > 2^{Cn/\Delta^{1/(k-2-\varepsilon)}}$.

Proof. Set $c = \varepsilon$ and $\delta = (\varepsilon' - c)/c$. By Lemma 2.22, there is a constant $C_{c,\delta} > 0$ such that with probability $1-o_n(1)$, G_F is an (r,c)=bipartite expander for $r = C_{c,\delta} \cdot n/\Delta^{1/(k-2-\varepsilon)}$ and hence, by Theorem 2.18, $\mathrm{width}(F) > cr/2$. Plugging this width lower bound (and k) into Theorem 2.10 yields the claimed bounds. □

In particular, we obtain the following

Corollary 2.24.
- *For any $c > 0$, random k-CNF formulas on n variables with at most cn clauses almost surely require resolution refutation size $2^{\Omega(n)}$.*
- *For every $\delta > 0$, random k-CNF formulas on n variables with at most $n^{k/2-\delta}$ clauses almost surely do not have polynomial-size resolution refutations.*

Proof. The first is an immediate implication. For the second, we have Δn clauses for $\Delta = n^{(k-2-2\delta)/2}$ in this case, for any $\delta > 0$, we have . can set $\varepsilon' = \delta$ and $\varepsilon = \delta/3$ in Theorem 2.23 to obtain a lower bound of the □

Even though boundary expansion of random formulas yields weaker width lower bounds for resolution proofs as the clause-variable ratio Δ grows with n, we show that random k-CNF formulas also have good boundary expansion if Δ is not too large. This is useful for the analysis of random formulas in other proof systems.

Lemma 2.25. *Let $\delta > 0$ and $0 < c < k-2-\delta c$. There is a constant $C'_{c,\delta} > 0$ such that if $\Delta = \Delta(n) \leq n^{\frac{(k-2-c)}{2}(1-\delta/2)}$ then for $F \sim \mathscr{F}_n^{k,\Delta n}$, G_F is an (r,c)-boundary expander with probability $1-o_n(1)$ for $r = C'_{c,\delta} \cdot k^{-1-2/(k-2-c)} \cdot n/\Delta^{2/(k-2-c)}$.*

Proof. Plug in $c' = (k+c)/2-1$ in place of c and $0 < \delta' = \delta\cdot(k-2-c)/[2(k+c-2)]$ in place of δ in Lemma 2.22. Observe that $k-2-c' = (k-2-c)/2 > \delta' c'$ and that $k-2-(1+\delta')c' = [(k-2-c)/2] - \delta' c' = \frac{(k-2-c)}{2}(1-\delta/2)$. Since $c' \geq (k-2)/2$ is already a function of k, rather than using $C = C_{c',\delta'}$ as the constant in the definition of r, we optimize the constant as a function of k. We chose C so that $C_{k,c'} \cdot C^{k-2-c'} \leq 1/2$, where

$$C_{k,c'} = e^{2+c'}(1+c')^{k-1-c'} = e^{(k+c+1)/2}[(k+c)/2]^{(k-c)/2} \leq e^{(k+c+2)/2}k^{(k-c)/2}.$$

Since $k-2-c' = (k-2-c)/2$, it suffices to choose

$$\begin{aligned} C &\leq 2^{-2/(k-2-c)} \cdot e^{-(1+(c+3)/(k-2-c))} \cdot k^{-1-2/(k-2-c)} \\ &\leq 2^{-2/(\delta c)} \cdot e^{-1-1/\delta-3/(\delta c)} \cdot k^{-1-2/(k-2-c)} \\ &= C'_{\delta,c} \cdot k^{-1-2/(k-2-c)}, \end{aligned}$$

for $C'_{\delta,c} = 2^{-2/(\delta c)} \cdot e^{-1-1/\delta-3/(\delta c)}$.

Therefore by Lemma 2.22, F is an (r, c')-bipartite expander with probability $1 - o_n(1)$. By Proposition 2.16, F is an (r, c)-boundary expander. □

2.5 Other applications and limitations of width-based lower bounds

In the general resolution lower bound of , the impact of the maximum input clause size of F from width(F) and the division by the number of variables can both limit the utility of the lower bound.

If we try to apply Theorem 2.10 to the PHP_n^{n+1} formula, we immediately run into problems, since PHP_n^{n+1} has $\Theta(n^2)$ variables, but width(PHP_n^{n+1}) is $O(n)$ so the largest lower bound from Theorem 2.10(b) is trivial. In addition, $w(PHP_n^{n+1}) = n$ which is similar to the width lower bound.

To get around this, one can define a restricted version of the pigeonhole principle, called the *graph pigeonhole principle*, $PHP(G)$, given any bipartite graph G on $[n+1]\times[n]$, which has variables x_{ij} only for $(i,j) \in E(G)$:

- Pigeon clauses: $\bigvee_{j:\ (i,j)\in E(G)} x_{ij}$ for each $i \in [n+1]$
- Hole clauses: $\neg x_{ij} \vee \neg x_{i'j}$ for all $j \in [n]$ and $i \neq i' \in [n+1]$ s.t. $(i,j),(i',j) \in E(G)$.

Then $PHP_n^{n+1} = PHP(K_{n+1,n})$ for the complete bipartite graph $K_{n+1,n}$. Moreover, $PHP(G)$ is a restriction of PHP_n^{n+1} in which all x_{ij} for $(i,j) \notin E(G)$ have been set to 0. Hence by Proposition 2.7, the restriction of any resolution refutation of PHP_n^{n+1} is a resolution refutation of $PHP(G)$ so it suffices to lower bound Res($PHP(G)$) for any graph G.

By similar calculations to the ones for random k-CNF formulas, if we choose G to be a random bipartite subgraph of $K_{n+1,n}$ that is 5-regular on the left, it is almost surely an (r,c)-expander for constant $c > 0$ and r that is linear in n.

A variant of the method allows us to prove lower bounds for Tseitin formulas on expanding graphs. Recall that there is a Tseitin formula $TS(G,\ell)$ for each undirected graph $G=(V,E)$ and each labeling function $\ell : V \to \{0,1\}$ with total parity *odd*; i.e., $\sum_{v\in V} \ell(v) \equiv 1 \pmod 2$. It has a variable x_e for each edge $e \in E$ and has:

- clauses representing the constraint that $\sum_{e\ni v} x_e \equiv \ell(v) \pmod 2$ for each $v \in V$.
  (If vertex v has degree d, there will be 2^{d-1} clauses representing the constraint at v.)

Definition 2.26. An undirected graph $G=(V,E)$ is an (r,c)-*edge expander* iff for every $S \subset V$ with $|S| \le r$, $|E(S, V-S)| \ge c|S|$ where $E(S,V-S)$ is the set of edges joining S and $V-S$.

Fact 2.27. *For every degree $d \ge 3$, there are undirected d-regular graphs $G=(V,E)$ on n vertices that are (an,c)-expanders for some constants $a,c>0$.*

It is not hard to see that if G is an (r,c)-edge expander then for any labelling function ℓ, the graph G_F for $F = TS(G,\ell)$ is an (r,c)-bipartite expander. However, it isn't a boundary expander because for each vertex $v \in V$ there 2^{d-1} clauses that have exactly the same edge variables and hence no boundary. However, since each input clause can be associated with a unique vertex, one can get a similar argument to go through by counting complexity of a clause C^* in a proof in terms of the size of the set of vertices of G whose clauses are used to derive C^*.

Rather than going through the details of this argument, in Chapter 3 we give a stronger argument that yields an exponential lower bound for PCR proofs and hence resolution proofs of expanding Tseitin formulas in Corollary 3.27.

Remark 2.28. A variant of the GT_n ordering formulas provides a counterexample to an asymptotic improvement (even for ordered resolution). These formulas have $N = n(n-1)$ variables and we have already seen that they have polynomial-size ordered resolution refutations. As originally defined the GT_n formulas have $w(GT_n) = n-1$ because of the long clauses stating that each $v \in [n]$ is not a minimal element. By adding $n(n-4)$ extension variables $y_{v,i}$ to express each long clause as an AND of 3-clauses (as in the usual reduction of CNF to 3-CNF that replaces a clause $(\ell_1 \vee \ldots \vee \ell_k$ by $(\ell_1 \vee \ell_2 \vee y_1), (\neg y_1 \vee \ell_3 \vee y_2), \ldots, (\neg y_{k-3} \vee x_{k-1} \vee x_k))$, we get a modified formula MGT_n, which has $w(MGT_n) = 3$ and clearly also has a polynomial-sized ordered resolution refutation. Bonet and Galesi![2] showed that MGT_n still requires resolution width at least $n-1$ which is essentially $\sqrt{N}$, despite its small resolution refutation size.

[2]

2.6 Resolution and Falsified Clause Search

While width does not provide us with a precise characterization of resolution complexity, there is another approach that does yield such a characterization, and has been convenient for other lower bounds.

Definition 2.29. Given an unsatisfiable CNF formula $F = \wedge_{i\in[m]} C_i$ in n variables, the *clause search* problem for F, Search_F is the following computational problem: On input $x \in \{0,1\}^n$, output an index $i \in [m]$ such that $C_i(x) = 0$.

Since F is unsatisfiable, Search_F is total; however Search_F is a relation rather than a function, since multiple clauses of F may be falsified by a given truth assignment x; any one of the answers allowed by the relation can be taken to be correct. Therefore we typically say “solve" rather than “compute" when describing algorithms that satisfy the conditions of Search_F.

Proposition 2.30. *DPLL on an unsatisfiable CNF formula F produces a decision tree that solves Search_F and hence $\text{Res}_{tree}(F)$ is equal to the size of the smallest decision tree solving Search_F.*

Proof. The decision tree for Search_F has the same form as the tree for DPLL (with the unit clause optimization ignored). Each node of the decision tree queries the underlying variable of the decision literal branched on in the DPLL tree and the output of the decision tree at a leaf is the index of an original clause that became empty at that leaf. The decision tree edge is labeled by the value of the variable consistent with it. □

Different subclasses of resolution correspond to variants of decision trees, known as *branching programs*. These are generalizations of decision trees that combine repeated common sub-trees into nodes of in-degree more than one; they have the structure of directed acyclic graphs (DAGs) and hence they can be thought of as *decision DAGs* or *decision diagrams*.

Definition 2.31. A (Boolean) *branching program (BP)* is a rooted directed acyclic graph in which

- each non-sink node is labeled by a variable (which is said to be *queried* at that node) and has out-degree 2
- the two out-edges from each non-sink node are labeled 0 and 1, and
- each sink node has an *output* label.

A branching program B computes a function of its input values in the same way that a decision tree does; that is, it starts at the root and at each node it evaluates the queried variable and follows the edge labeled by that value to the next node. Its output value is the label of the sink reached. The *size* of

a branching program is the number of its nodes, and its *length* is the length of the longest path from the root to any sink.

A branching program is said to be *read-once* if no variable is queried more than once on any path from the root to a sink. It is said to be an *ordered binary decision diagram (OBDD)* if it is read once and the order on which variables are read on every root-sink path is the same.

Example 2.32. Branching programs, even ODBBs, can be much smaller than decision trees (which correspond to the special case that the DAG is a tree). The simplest example illustrating this is the n-bit parity function $\oplus_n$: While any decision tree for $\oplus_n$ requires 2^n leaves (since no path can stop before reading all n input bits), there is an OBDD of size $2n+1$ computing $\oplus_n$ that queries the n bits in order and records only whether the parity so far is even or odd.

OBDDs have been widely used as data structures to represent Boolean functions (particularly for verification prior to the development of CDCL SAT solvers). Branching programs have been used in many areas of computational complexity in part because the size and length measures correspond naturally to space and time measures simultaneously. (While length corresponds directly to time, it is log of size that corresponds to space.)

Remark 2.33. At each node v of a read-once branching program B, we can associate the set of variables S_v that have been read (queried) on some path from the root to v. By the read-once property, none of the variables in S_v can be read at any node reachable from v. By at most a factor n increase in size we can modify B so that for every node v, the variables in S_v are read along *every* path from the root v. We can think of the name of node v as telling us everything we need to know about the values of the variables in S_v that we need in order to compute the answer.

Theorem 2.34. *Given an unsatisfiable CNF formula F,*

- $\mathrm{Res}_{reg}(F)$ *equals the size of the smallest read-once branching program solving* $Search_F$.
- $\mathrm{Res}_{order}(F)$ *equals the size of the smallest OBDD solving* $Search_F$.

Proof. Suppose that R is a regular resolution refutation of size S for F. Each clause C appearing in R is a node of B. If two clauses $A \vee x$ and $B \vee \neg x$ in R resolve on a variable x to produce the clause C, then in the branching program B we branch from the node C on the variable x to reach $A \vee x$ on the $x = 0$ branch, and $B \vee \neg x$ on the $x = 1$ branch. This maintains the property that all assignments that reach a node v of B falsify the clause labeling v. The resulting branching program B solves the conflict clause search problem for F and has the same size as the refutation R. The fact that no variable is branched on more than once on any path is immediate

from the definition; the fact that this results in an OBDD in the case of ordered resolution is also immediate.

In the other direction, we obtain a regular refutation R from the read-once branching program B. We will label each node v with the maximal clause C_v that is falsified by every assignment reaching v. These clauses form the regular resolution refutation. If v is a leaf then C_v is the conflicting clause from F found by B. If B branches from node v on a variable x to nodes v_0, v_1, then in R we resolve the clauses C_{v_0}, C_{v_1} on x to obtain C_v. Again, the number of clauses in the refutation R is the same as the number of nodes in the branching program B.

The fact that the resolution refutation is regular follows immediately from the fact that the branching program is read-once; if the branching program is an OBDD then it is immediate that the resolution refutation is ordered. □

Because these are characterizations, they work both for finding efficient proofs and for proving lower bounds.

For example of the former, we can consider the formulas $PEB(G)^{(\oplus 2)}$ in which we modify the pebbling formula for a graph G by replacing each variable x_v by the XOR of two variables $y_v \oplus z_v$ and expanding the result as clauses. We will prove the following proposition when we discuss lifting.

Theorem 2.35. *There is a constant $c > 0$ such that for any rooted directed acyclic graph G, $\mathrm{Res}_{tree}(PEB(G)^{(\oplus 2)}) \geq 2^{c \cdot \mathrm{peb}(G)}$.*

We can use our characterization to prove that all such formulas have small regular resolution refutations:

Theorem 2.36. *For any rooted directed acyclic graph G, $F = PEB(G)^{(\oplus 2)}$ has a polynomial size regular resolution refutation.*

Proof. We show this via a read-once branching program B for Search_F. We describe B as if it were an algorithm and then describe the nodes of B as a branching program. On input (y, z), B begins at the root (sink) vertex t of G. and queries the variables y_t and z_t. If their parity is 1, then B can identify a violated clause and halt. For each of the remaining steps, we assume that B has identified a vertex v of B and queried variables y_v and z_v whose parity is 0 (and has forgotten the values of any other variables). If v is a source, then B can output a violated axiom. Otherwise, B will query the values of the predecessors of v (that is query the values of y_v and z_v in turn. For each predecessor u, if their parity is 0, B will move attention to u and forget all information about v. If their parity is 1, it will move to the next predecessor of v. If all predecessors of v have value 1, then B can output the index of a violated clause defined on the y and z values for v and its predecessors.

As a branching program, for each vertex v of G, B will have one node n_v^0 for the parity 0 assignments $y_v = z_v = 0$ and one node n_v^1 for the parity

0 assignment $y_z = z_v = 1$. If v is a source, then each of n_v^0 and n_v^1 will be labeled by a violated source clause. Each of these two nodes will be followed by a decision tree on the values of the y_u and z_u for each of the predecessors u of v (with the y_u read first). If an edge for z_u such that $y_u \oplus z_u = 0$ will leave this decision tree and go to node $n_u^{z_u}$. Each leaf of the decision tree that remains (which corresponds to the case that all predecessors u have $y_u \oplus z_u = 1$) is labeled by the associated violated clause. Finally, the root of B will have a height two decision tree on y_t and z_t; two of its leaves will be labeled by the violated sink clause and the other two will be n_t^0 and n_t^1. The total size is $O(|F|)$. □

Using the fact that there are graphs G with pebbling number $\Omega(n/\log n)$ we obtain a separation.

Corollary 2.37. *Tree resolution proof size can be exponentially larger than regular resolution proof size.*

Techniques for proving lower bounds for computations of functions using read-once branching programs have been adapted for use with the falsified clause search problem and can make it easier to prove lower bounds for regular resolution than for general resolution. However, there are relatively few kinds of instances known in which there is a provable separation.

Proposition 2.38. *There are CNF formulas F in n variables that have $n^{O(1)}$ size resolution refutations but require regular resolution refutations of size $2^{n^{\Omega(1)}}$.*

One example of such an instance is the following modification of the GT_n instances: Fix some tricky function $f : [n]^{(3)} \to [n]^{(2)}$ taking ordered triples of distinct indices to ordered pairs of distinct indices and replace each of the transitivity axioms $\neg x_{uv} \vee \neg x_{vw} \vee x_{uv}$ by two axioms

$$\neg x_{uv} \vee \neg x_{vw} \vee x_{uv} \vee x_{f(u,v,w)}$$

and

$$\neg x_{uv} \vee \neg x_{vw} \vee x_{uv} \vee \neg x_{f(u,v,w)}.$$

Clearly, an ordinary resolution proof can first resolve each of these pairs to recover the original GT_n formula and then prove GT_n in polynomial size as before. This proof will not be regular because that proof will encounter the variable $x_{f(u,v,w)}$ again later as part of its variable elimination for GT_n. For certain functions f, one can show that it is not possible to find an efficient regular resolution refutation.

Given that the sub-classes of resolution proofs correspond perfectly to restricted branching programs it would be natural to guess that general branching programs solving Search_F would correspond to general unrestricted resolution, but that is not the case. Such a model would be too

powerful: In particular, a general branching program could simply evaluate the clauses one-by-one until it reaches a clause that is falsified. This would read each variable many times but would essentially have size only as large as the number of literals in the input formula F.

Instead, the computational model for which resolution solves Search_F are *cube DAG protocols*. Such protocols, like decision trees, remember a partial assignment as the computation proceeds. Protocols typically correspond to trees but here the structure of the computation is a DAG.

Definition 2.39. A *cube DAG protocol* for a relation R from $\{0,1\}^n$ to $[m]$ is a rooted directed acyclic graph (DAG) that has out-degree 2 at non-sink nodes. Each node v of the DAG is labeled by a sub-cube of the Boolean cube that is the set of elements $C_v \subseteq \{0,1\}^n$ consistent with a fixed partial assignment α_v. We have the following other requirements:

(a) The root is labeled by the full cube $\{0,1\}^n$ corresponding to the empty partial assignment.

(b) For each node u of the DAG with children v, w, we have $C_u \subseteq C_v \cup C_w$

(c) Each sink node s is also labeled by an *output* value i such that $C_s \subseteq R^{-1}(i)$. That is, $(x,i) \in R$ for all $x \in C_s$.

On an input $x \in \{0,1\}^n$, a cube DAG protocol starts at the root; at each node u, it can move to child node v if $x \in C_v$. (Note that when $x \in C_v \cap C_w$, we assume that it moves to C_v.) The protocol outputs the value of a leaf reached.

Theorem 2.40. *For an unsatisfiable F, $\text{Res}(F)$ is the size of a smallest cube DAG protocol for Search_F.*

Proof. We obtain a 1-1 correspondence between resolution refutations of F and cube DAG protocols for Search_F, by using a 1-1 correspondence between clauses and the sets of total assignments that make them false (which by definition are sub-cubes). The length of the clause is the length of the longest partial assignment that all these total assignments share. We can apply this correspondence in either direction.

In particular, the root, which is labeled by the empty clause in the resolution proof, is labeled by the associated sub-cube consisting of all inputs in the cube DAG protocol. It is clear by soundness that any assignment falsifying $A \vee B$ must falsify one of $A \vee x$ and $B \vee \neg x$ so the cube property (b) holds. This implies that a resolution refutation directly yields a cube DAG protocol.

For the other direction, we first assume without loss of generality that the cube DAG protocol is *minimal* in the sense that no nodes can be removed and no cube C_v can be locally replaced by a smaller cube that retains the cube DAG protocol property.

In particular, a minimal cube DAG protocol cannot have any nodes u with children v and w such that $C_u \subseteq C_v$ or $C_u \subseteq C_w$.

If we have two sub-cubes C_v and C_w in this proof whose union covers C_u, then the defining assignments α_v and α_w must be subsets of the defining assignment α_u for C_u, except for the value of one variable x that has opposite signs in the two assignments α_v and α_w. If $\text{supp}(\alpha_v) \cup \text{supp}(\alpha_w) \subsetneq \text{supp}(\alpha_u) \cup \{x\}$ then the protocol is not minimal since we can reduce C_v (or C_w) by adding in the remaining assignments from α_u into α_v (respectively α_w). Therefore we can assume that this does not happen at any node u, and we easily obtain that the cube DAG proof step exactly matches a resolution step. □

Lower bounds for general resolution proofs for which the width and restriction-based methods fail are often attacked using the following class of strategies called Prover-Adversary games:

Definition 2.41. A *prover-adversary game* for a CNF formula F is a game with two players having conflicting goals, the Prover and the Adversary. The Adversary claims that F has a satisfying assignment, while the Prover tries to catch the Adversary in a lie. The Prover maintains a partial assignment α to the n variables, initially empty, that is known to the Adversary. The Prover receives these values by asking the Adversary but the Adversary may lie and give inconsistent answers when the same question is asked again for a variable that is not in α at the time. At each step

- The Prover selects an input position i (based only on α and asks the Adversary for the value of x_i
- The Adversary chooses a value $b_{i,t} \in \{0,1\}$ and tells the Prover that $x_i = b_{i,t}$.
- The Prover adds $x_i = b_{i,t}$ to the current assignment α and removes some portion of the assignment α that can be forgotten.

The game ends when the Prover has a partial assignment that falsifies some clause of F.

A Prover strategy in this case is a method for determining which index i to ask for given the current values of the partial assignment α. The complexity of the game is the minimum over all Prover strategies of the number of different configurations α that the Prover might need to remember given different actions by the Adversary.

Proposition 2.42. *The complexity of the Prover-Adversary game for unsatisfiable CNF formula F is its resolution complexity.*

It should be clear that cube DAG protocols precisely correspond to the possible executions of Prover-Adversary games and there is one node in

those protocols for each configuration α that the Prover can be forced to maintain by the Adversary.

One lower bound that cannot be proven by the width-based methods is for the grid Tseitin formulas, namely Tseitin formulas on the nxn grid graph. This formula has $O(n^2)$ variables but requires proof width only n (for example, there are cuts of size n that split the graph in two). Therefore, the general resolution lower bound of Theorem 2.10 is trivial. Nonetheless Dantchev and Riis proved the following lower bound using the Prover-Adversary game:

Theorem 2.43. *The nxn grid Tseitin formulas require resolution complexity $2^{\Omega(n)}$.*

3
The Complexity of Algebraic Proofs

Non-clausal translations When we previously described the versions of the pigeonhole principle we assumed that we used a standard translation of the long clauses, resulting in an input representation that was already high degree. We have the following natural algebraic version of PHP_n^m for algebraic proofs over any field:

- Pigeon equations: $\sum_{j\in[n]} x_{ij} - 1 = 0$ for all $i \in [m]$,
- Hole equations: $x_{ij} \cdot x_{i'j} = 0$ for all $i \neq i' \in [m]$ and $j \in [n]$.

This is somewhat more restrictive semantically than the clausal version of PHP_n^m but it is more natural in this context.

Note that over a field of non-zero characteristic p, the Pigeon equations are not sufficient semantically to imply the Functional equations since they would only require that each pigeon map to a number of holes that is $\equiv 1 \pmod{p}$. For the $function\text{-}PHP_n^m$, we would add

- Functional equations: $x_{ij} \cdot x_{ij'} = 0$ for all $i \in [m]$, and $j \neq j' \in [n]$,

For the $bijective\text{-}PHP_n^m$ we can also add:

- Surjectivity equations: $\sum_{i\in[m]} x_{ij} - 1 = 0$ for all $j \in [n]$.

3.1 Nullstellensatz Proofs and Counting

If we have both the Pigeon equations and Surjectivity equations, then we can simply add them up with opposite signs to yield the following:

Proposition 3.1. *The algebraic version of $bijective\text{-}PHP_n^{n+1}$ has a Nullstellensatz refutation of degree 1 over any field.*

It is easy to see that the same holds for PHP_n^m provided that $m-n$ is not divisible by the field characteristic p.

The modular counting formulas $COUNT_k^n$ when n is not divisible k, stating that $[n]$ cannot be partitioned into sets of size k also have nice algebraic forms using variables x_e for $e \in \binom{[n]}{k}$, we have:

- $\sum_{e \ni v} x_e - 1 = 0$ for all $v \in [n]$
- $x_e \cdot x_f = 0$ for all $e \perp f$ (which means $\emptyset \neq e \cap f \neq e$).

Again, by adding up, we have the following:

Proposition 3.2. *For prime p, $COUNT_p^{pn+1}$ has degree 1 Nullstellensatz refutations in fields of characteristic p.*

We will see that over fields $\mathbb{F}$ of characteristic $q \neq p$, $COUNT_p^{pn+1}$ requires refutations of linear degree even in $\mathbf{PC}_{\mathbb{F}}$.

3.2 Nullstellensatz Degree Lower Bounds via Designs

Definition 3.3. A *d-design* for vector of multilinear polynomials $\vec{f} = f_1, \ldots, f_m$ in $\mathbb{F}[x_1, \ldots, x_n]$ is a linear mapping $D : \mathbb{F}^{\leq d}[x_1, \cdots, x_n] \to \mathbb{F}$ with the following properties:

- $D(1) = 1$
- For every polynomial $g \in \mathbb{F}[x_1, \ldots, x_n]$ with the degree of $f_i \cdot g$ at most d, $D(f_i \cdot g) = 0$.

Since D is linear, it is determined by its values on the monomials $\prod_{i \in S} x_i$ for $|S| \leq d$. since we are working modulo the ideal I which reduces all polynomials to multilinear polynomials, we can think of D as a function $D : \binom{[n]}{\leq d} \to \mathbb{F}$ on subsets of indices of size at most d, providing a "fake" evaluation in $\mathbb{F}$ of the monomial $x_S = \prod_{i \in S} x_i$.

A d-design for $\vec{f}$ provides an impediment to Nullstellensatz refutations.

Proposition 3.4. *There is a d-design for $\vec{f}$ over $\mathbb{F}$ if and only if $\vec{f}$ requires Nullstellensatz refutations of degree at least $d + 1$.*

Proof. We simply apply a d-design D to both sides of a purported Nullstellensatz derivation. By the first condition, the right hand-side evaluates to 1, but by the second condition, the left-hand side evaluates to 0. This contradicts the claimed equality between the two. □

Like Nullstellensatz proofs themselves, and because d-designs are determined by their values on monomials, we can view the constraints for a d-designs as systems of linear equations in the $D(S)$. In particular, it suffices to add constraints on the various $D(S)$ to ensure that $D(\prod_{j \in T} x_j \cdot f_i) = 0$ for each T of size at most $d - \deg(f_i)$ and $i \in [m]$.

Using designs, one can prove the following:

Proposition 3.5. *The induction principle IND_n can be proven in Nullstellensatz degree $\lceil \log_2 n \rceil$ and this is optimal.*

Proof Sketch. For the upper bound, we observe that the algebraic translation of IND_n has polynomials $1 - x_1 = 0$, $x_i(1 - x_{i+1}) = 0$ for $i \in [n-1]$ and $x_n = 0$. For simplicity assume that $n = 2^k + 1$ for some integer $k \geq 0$.

We use the simulation of tree-resolution height (DPLL depth) by Nullstellensatz proofs for the upper bound. If $k = 0$ we get an immediate violation of degree 2. Otherwise, let $m = (n+1)/2 = 2^{k-1} + 1$. The DPLL queries x_m; if $x_m = 0$ then the DPLL algorithm is run recursively on the IND_m sequence $x_1, \ldots, x_m$; if $x_m = 1$ then the DPLL algorithm is run recursively on the sequence $x_m, \ldots, x_n$, which also has length $m = 2^{k-1} + 1$.

We prove the design-based lower bounds in a more general context below. □

This is a special case of a more general result about the Nullstellensatz degree and size required for refuting the pebbling formulas $PEB(G)$ for single-sink directed acyclic graphs G. The lower bound is based on the *reversible pebble game* on G, which was originally defined in order to understand how efficiently general computation can be simulated by reversible computation (and as a result has had applications to quantum computing).

Definition 3.6. The *reversible pebble game* on single-sink directed acyclic graph G is played according to the following rules:

- If all predecessors of a node v have a pebble and v does not, then a pebble may be placed on v.
- If all predecessors of a node v have a pebble and v has a pebble, then that pebbled may be removed from v.

The game starts with no pebbles on G, must reach a step when there is a pebble on the sink of G and must end with an empty graph. (Observe that the steps of the second half can be the reverse of the first half without loss of generality.)

The time of a game is the # of moves, and the space is the maximum number of pebbles that are on G at any point during the game. The *reversible pebble number* of G is the minimum space required for any successful strategy.

Recall that $IND_n = PEB(P_n)$ where P_n is a directed path of n vertices.

Proposition 3.7. *The reversible pebble number of P_n is precisely $\lceil \log_2(n+1) \rceil$.*

Proof. The upper bound is natural: Number the vertices of P_n by $[n]$. Let $k(p)$ be the furthest vertex one can reversibly pebble in the path with p pebbles.

CLAIM: $k(p) = 2^p - 1$

Clearly $k(1) = 1$. We prove that $k(p+1) = 2k(p) + 1$ for all $p \geq 1$, which proves the claim.

For the pebbling strategy with $p+1$ pebbles, run the strategy for p pebbles until a pebble is placed on vertex $k(p)$, then place the $(p+1)^{\text{st}}$ pebble on vertex $k(p)+1$. and then run the rest of the p-pebble strategy to remove the first p pebbles for re-use. With the pebble on $k(p)+1$, these p pebbles can be used to pebble the next $k(p)$ nodes reaching node $2k(p)+1$. Therefore $k(p+1) \geq 2k(p)+1$.

For the optimality of this strategy, observe that in any pebbling strategy with $p+1$ pebbles, the smallest numbered vertex containing a pebble must be at most $k(p)+1$, because otherwise such a pebble can never be removed because there are only p other pebbles available, which can reach $k(p)$ at most. The other p pebbles can then reach at most $k(p)$ further. □

Theorem 3.8. *G has a reversible pebbling strategy of time $2T$ and space S if and only if $PEB(G)$ has a Nullstellensatz refutation (over any field) of degree S and size $T+1$. In particular, $PEB(G)$ requires Nullstellensatz degree equal to the reversible pebbling number of G.*

Proof. We first describe the easy direction. Observe that the translation of $PEB(G)$ has polynomials of the form: $Q_v = \prod_{u \in pred(v)} x_u \cdot (1 - x_v)$ for every vertex v, and $Q'_t = x_t$ where t is the sink. Let $\emptyset = S_0, \ldots, S_T, \ldots, S_{2T} = \emptyset$ be the sequence of sets of vertices occupied by the pebbles in a reversible pebbling strategy on G. For each set S_i, let the monomial $x_{S_i} = \prod_{w \in S_i} x_w$. Observe that $x_{S_0} = 1$. We claim that $x_{S_{i+1}} - x_{S_i}$ is simply Q_v multiplied by a (signed) monomial. By definition $S_{i+1} = S_i \oplus \{v\}$ for some vertex v where $pred(v) \subseteq S_i, S_{i+1}$. Let $A_i = S_{i+1} \setminus (\{v\} \cup pred(v))$ and assume w.l.o.g. that $v \in S_i$. Then $x_{S_{i+1}} - x_{S_i} = x_A \cdot \prod_{u \in pred(v)} x_u - x_A \cdot \prod_{u \in pred(v)} x_u \cdot x_v = x_A \cdot Q_v$. Also since $t \in S_T$, we define $B = S_T - \{t\}$ and observe that $x_{S_T} = x_B \cdot x_t = x_B \cdot Q'_t$. Writing out the terms, we get that the right hand side of the telescoping sum:

$$0 = x_{S_0} + (x_{S_1} - x_{S_0}) + (x_{S_2} - x_{S_1}) + \cdots + (x_{S_{t-1}} - x_{S_T}) + x_{S_T}$$

can be written as 1 plus a sum of T terms of the form $x_A \cdot Q_v$ or $x_B \cdot Q'_t$. The degree of the proof is the size of the largest set S_i.

For the other direction, we use a d-design to prove the degree lower bound and omit the more general simulation, which is more involved. Suppose that d is strictly smaller than the reversible pebbling number of G. For $U \subset V$ with $|U| \leq d$, define $D(U) = 1$ iff U is a set of nodes that is a configuration of pebbles that can be reached starting from the empty graph. and $D(U) = 0$ otherwise.

Since $\emptyset$ is reachable, we have $D(\emptyset) = 1$ as required. Since d is strictly smaller than the pebbling number of G, we have $D(U) = 0$ for any U containing the sink t, $D(x_B \cdot Q'_t) = 0$ for all B of size at most $d-1$. Finally, observe that for any A of size at most $d - (pred(v)+1)$, $x_A \cdot Q_v = x_A \cdot x_{pred(v)}) - x_A \cdot x_{pred(v)} \cdot x_v = x_{A \cup pred(v)} - x_{A \cup pred(v) \cup \{v\}}$. Now $A \cup pred(v)$

is reachable if and only if $A \cup pred(v) \cup \{v\}$ is reachable (this is where we required reversibility), so $D(x_A \cdot Q_v) = 0$. Therefore we have met the conditions for a d-design and hence degree must be at least the reversible pebbling number, as required. □

Since doing reversible pebbling requires at least as many pebbles as ordinary pebbling when pebbles can be removed freely, we have the following fact:

Proposition 3.9. *There is a family of graphs for each n, with n vertices, constant in-degree at most k, and reversible pebbling number $\Omega(n/\log n)$.*

This yields the following nearly-optimal separation between the degree of Nullstellensatz and $\mathbf{PC}_{\mathbb{F}}$ proofs.

Corollary 3.10. *The family of graphs G in Proposition 3.9 have $\mathbf{PC}_{\mathbb{F}}$ refutations of constant degree $k+1$ but require Nullstellensatz degree $\Omega(n/\log n)$ over any field.*

Proof. The lower bound is immediate from Theorem 3.8. For the upper bound observe that the DPLL unit propagation derivation yields a tree-resolution refutation where each resolution inference involves one clause of size 1 and hence the clauses never are larger than the input clauses, which are of size at most $k+1$. The PCR simulation of resolution has degree equal to the clause size and the $\mathbf{PC}_{\mathbb{F}}$ degree is the same as that of PCR. □

3.3 Size versus Degree for Polynomial Calculus and PCR proofs

In the simulation of resolution by PCR proofs in Theorem 1.24, the degree of the proof is precisely the width of the resolution proof (and hence the same applies to PC degree). We will see that for PC and PCR proofs, there is a simulation of small proofs by low degree proofs, that is exactly analogous to the one we proved in Theorem 2.8 converting small size resolution proofs to low width proofs. In fact, the version for PC was proven prior to the one for resolution. This will allow us to derive size lower bounds for PC proofs (and, more interestingly, PCR proofs) by proving degree lower bounds on such proofs.

Recall our notation that we write $f_1, \ldots, f_m \vdash_d f$ iff there is a degree d $\mathbf{PC}_{\mathbb{F}}$ (and hence $\mathbf{PCR}_{\mathbb{F}}$) derivation of f from $f_1, \ldots, f_m$.

Theorem 3.11. *Let F be a CNF formula in n variables. If $\mathbf{PCR}_{\mathbb{F}}(F) \leq S$ or $\mathbf{PC}_{\mathbb{F}}(F) \leq S^2$ then $\deg(F) \leq \max(2\sqrt{2n \ln S}, \sqrt{2n \ln S} + w(F))$.*

Proof. We prove the statement for $\mathbf{PCR}_{\mathbb{F}}$. Set $D = \lceil \sqrt{2n \ln S} \rceil$. We prove the following claim by induction on n and k:

CLAIM: If $(1 - D/2n)^k \, S < 1$ then any CNF formula F in n variables having a $\mathbf{PCR}_{\mathbb{F}}$ refutation (derivation of 1) with $\leq S$ monomials of degree at has

$\deg(F) \leq \max(D, w(F)) + k$.

Before proving the claim, we observe that the case $k = D$ is sufficient to prove the theorem, since $(1 - D/2n)^D < e^{-D^2/2n} \leq 1/S$ by the choice of D.

The case $k = 0$ is trivial.

For the general case, let P be a $\mathbf{PCR}_{\mathbb{F}}$ refutation of F with n variables and $\leq S$ monomials of degree $\geq D$ and suppose that $(1 - D/2n)^k \; S \leq 1$. Write $\overline{p}(F)$ be shorthand for the polynomial translation of F. Let $d = \max(D, w(F)) + k$ Choose the variable z appearing in the most wide clauses of P. Since there are at most $2n$ possible $\mathbf{PCR}_{\mathbb{F}}$ variables and $\geq D$ distinct variables per high degree monomial, z appears in $\geq DS/2n$ monomials of degree $\geq D$.[1]

[1] For $\mathbf{PC}_{\mathbb{F}}$, there are only n variables instead of $2n$ so $DS/2n$ is replaced by DS/n in the case of $\mathbf{PC}_{\mathbb{F}}$. This is the only difference in the argument.

Consider the restrictions $z \leftarrow 0$ and $z \leftarrow 1$: Then $P_{z \leftarrow 0}$ is a $\mathbf{PCR}_{\mathbb{F}}$ refutation of $F_{z \leftarrow 0}$ and every monomial of P containing z is set to 0 (and others are only shortened), so $P_{z \leftarrow 0}$ has $S' \leq S - DS/2n = (1 - D/2n)\, S$ monomials of degree $\geq D$. Therefore $(1 - D/2n)^{k-1} \; S' \leq (1 - D/2n)^k \; S \leq 1$. It follows by our inductive hypothesis with $k - 1$, that

$$\deg(F_{z \leftarrow 0}) \leq \max(D, w(F_{z \leftarrow 0})) + k - 1 \leq \max(D, w(F)) + k - 1 = d - 1;$$

that is, $\overline{p}(F_{z \leftarrow 0}) \vdash_{d-1} 1$. Therefore, by multiplying lines of this proof by z as needed, we get $\overline{(p)}(F) \vdash_d z$.

Now, $P_{z \leftarrow 1}$ is a $\mathbf{PCR}_{\mathbb{F}}$ refutation of $F_{z \leftarrow 1}$. By the inductive hypothesis applied to $F_{z \leftarrow 1}$, which has $n' = n - 1$ variables, there is a $\mathbf{PCR}_{\mathbb{F}}$ refutation P'' of $F_{z \leftarrow 1}$ of degree at most $\max(D, w(F)) + k = d$; that is, $\overline{p}(F_{z \leftarrow 1}) \vdash_d 1$.

Observe that the clauses C of F that are shortened by $z \leftarrow 1$ contain $\neg z$ and hence their polynomials $\overline{p}_C$ are multiples of z, which we have already derived. Hence, since $d \geq w(F)$, putting things together we have

$$\overline{p}(F) \vdash_d \overline{p}(F), z \vdash_d \overline{p}(F_{z \leftarrow 1}) \vdash_d 1$$

as required. □

Corollary 3.12. *For any field $\mathbb{F}$ and for any CNF formula F, $\mathbf{PC}_{\mathbb{F}}(F) \geq 2^{(\deg(F) - w(F))^2/(2n)}$ and $\mathbf{PCR}_{\mathbb{F}}(F) \geq 2^{(\deg(F) - w(F))^2/(8n)}$.*

3.4 Gaussian Width and Polynomial Calculus over the Fourier (± 1) basis

We will take a digression to a notion of derivations and refutations for unsatisfiable systems of linear equations.

Definition 3.13. A *Gaussian derivation* of $a^T x = b$ for $x \in \mathbb{F}^n$ from constraints $a_1^T x = b_1, \ldots a_m^T x = b_m$ is a sequence of equations $a_1^1 x = b_1$, …, $a_S^T x = b_S$ where $a = a_S$, $b = b_S$ and for $j > m$, there exist $i, i' < j$ and constants $c, d \in \mathbb{F}$ such that $a_j = c \cdot a_i + d \cdot a_{i'}$ and $b_j = c \cdot b_i + d \cdot b_{i'}$. If

this is a derivation of $0 = 1$, i.e. $a = 0$ and $b = 1$, then this is a *Gaussian refutation* of the constraints.

Obviously, Gaussian elimination yields a Gaussian derivation, which is the reason for the terminology. For n-dimensional variables, Gaussian derivations can always be short, and only require n^2 steps, but we will be interested in a different parameter.

Definition 3.14. The *width* of set $\mathscr{L}$ of linear equations of the form $a^T x = b$, $w(\mathscr{L}) = \max_{a^T x = b \in \mathscr{L}} |a|_0$, the maximum number of variables appearing in any equation in $\mathscr{L}$. In particular, we can define he *Gaussian width* $w_G(\mathscr{L})$ of a set of equations $\mathscr{L}$ to be the minimum width of any Gaussian refutation of $\mathscr{L}$ if $\mathscr{L}$ is unsatisfiable, and ∞ otherwise.

Definition 3.15. For a system $\mathscr{L}$ of linear equations, we define a bipartite graph $G_{\mathscr{L}} = (\mathscr{L}, [n], E)$ where $(\ell, i) \in E$ iff ℓ is $a^T x = b$ with $a_i \neq 0$.

Proposition 3.16. *Let $\mathscr{L}_G^\ell$ be the set of parity equations associated with Tseitin formula $TS(G, \ell)$. If G is an (r, c)-edge expander, then $\mathscr{L} = \mathscr{L}_G^\ell$ is an unsatisfiable set of equations and $G_{\mathscr{L}}$ is an (r, c)-boundary expander.*

Proof. This is practically by definition: there is precisely one equation in $\mathscr{L}_G^\ell$ for each vertex in G and the edge boundary of a set of vertices S is precisely the vertex boundary of the equations indexed by S in $G_{\mathscr{L}_G^\ell}$. □

Lemma 3.17. *If $G_{\mathscr{L}}$ is an (r, c)-boundary expander, then $w_G(\mathscr{L}) > cr/2$.*

Proof. We can assume that $c > 0$. Since $G_{\mathscr{L}}$ is an (r, c)-boundary expander, any set of at most r equations is satisfiable. Therefore a refutation of $\mathscr{L}$ must involve an equation ℓ that is a linear combination of $> r$ equations of $\mathscr{L}$. Since each line in a Gaussian derivation is a linear combination of 2 previous lines, there is a line ℓ^* of the Gaussian refutation that is a linear combination of set $S \subset \mathscr{L}$ with $r/2 < |S| \leq r$. The $c|S| > cr/2$ boundary variables associated with S must all appear in ℓ^*. □

In order to prove degree lower bounds for polynomial calculus and PCR over fields $\mathbb{F}$ of characteristic other than 2, it is convenient to consider algebraic proof systems in which algebraic variables $z_1, \ldots, z_n$ are assigned algebraic values $z_i = (-1)^b \in \mathbb{F}$ corresponding to the logical assignment $x_i = b$.

Definition 3.18. We write $\mathbf{PC}_{\mathbb{F}}^{\pm}$ for the variant of $\mathbf{PC}_{\mathbb{F}}$ in which we work modulo the ideal $I_{\pm 1}$ generated by the polynomials $z_i^2 - 1$ rather than the ideal $I_{\{0,1\}}$ generated by the polynomials $x_i^2 - x_i$.

In $\mathbf{PC}_{\mathbb{F}}^{\pm}$ we represent a clause $C = \bigvee_{i \in P} x_i \vee \bigvee_{j \in N} \overline{x_j}$ as $p'_C = \prod_{i \in P} \frac{1+z_i}{2} \cdot \prod_{j \in N} \frac{1-z_j}{2}$.

We can compare p'_C with the representation $p_C = \prod_{i \in P}(1 - x_i) \cdot \prod_{j \in} x_j$ of clause C in $\mathbf{PC}_{\mathbb{F}}$ without dual variables or $\overline{p}_C = \prod_{i \in P} \overline{x}_i \cdot \prod_{j \in N} x_j$ with dual variables.

Definition 3.19. Suppose that $\mathrm{char}(\mathbb{F}) \neq 2$ and define the multilinear mapping $\phi : \mathbb{F}[x_1, \ldots, x_n] \to \mathbb{F}[z_1, \ldots, z_n]$ that replaces each x_i by $(1 - z_i)/2$. Its inverse, ϕ^{-1} is the multilinear map that replaces each z_i by $1 - 2x_i$.

Observe that $\phi(x_i^2 - x_i) = (z_i^2 - 1)/4$ and that if p_C is the representation of a clause C in $\mathbf{PC}_{\mathbb{F}}$ then $p'_C = \phi(p_C)$ is its representation in $\mathbf{PC}^{\pm}_{\mathbb{F}}$.

Proposition 3.20. *Suppose that* $\mathrm{char}(\mathbb{F}) \neq 2$. *Let* $f_1, \ldots, f_m, f \in \mathbb{F}[x_1, \ldots, x_n]$. *Then the degree of inferring* f *from* $f_1, \ldots, f_m$ *in* $\mathbf{PC}_{\mathbb{F}}$ *is the same as the degree of inferring* $\phi(f)$ *from* $\phi(f_1), \ldots, \phi(f_m)$ *in* $\mathbf{PC}^{\pm}_{\mathbb{F}}$.

Proof. Apply ϕ to each line of the derivation of f to derive the proof of $\phi(f)$. (Each $x_i^2 - x_i$ defining the ideal is converted to $(z_i^2 - 1)/4$ which is equivalent to $z_i^2 - 1$ since $\mathrm{char}(\mathbb{F}) \neq 2$.) The map ϕ preserves the degree since it is multilinear. In the reverse direction, the map ϕ^{-1} also preserves the degree. □

This ± 1 basis is sometimes called the Fourier basis because it is the natural basis for Fourier analysis of Boolean functions over $\mathbb{F}$. It can be much more efficient with respect to size for some properties, since a single monomial in this basis represents a parity function. This makes it particularly convenient in representing linear equations over field $\mathbb{F}_2$. In particular, the following lemma is immediate from that fact that $\phi(x_i) = (-1)^{x_i}$:

Proposition 3.21. *Let* $S, T \subseteq [n]$. *For* $z_i = \phi(x_i)$ *for* $i \in [n]$,

$$\prod_{i \in S} z_i^{a_i} = (-1)^c \cdot \prod_{i \in T} z_i^{b_i} \tag{3.1}$$

if and only if

$$\sum_{i \in S} a_i \cdot x_i = c + \sum_{i \in T} b_i \cdot x_i$$

over $\mathbb{F}_2$.

Equations of the form Eq. (3.1), are expressed by setting *binomial* polynomials $p = M_1 \pm M_2 = 0$ over $\mathbb{F}_2$ where M_1 and M_2 are monomials. For a binomial p, we use ℓ_p to denote the linear equation given by Proposition 3.21 that expresses $p = 0$. Note that this representation allows for multiple versions of the same constraint since the same variable may appear on both left and right sides (in both S and T). The width of this equation is at most $|S| + |T|$. The following lemma gives a special property of $\mathbf{PC}^{\pm}_{\mathbb{F}}$ proofs when the input polynomials are all binomials.

Theorem 3.22. *If there is a* $\mathbf{PC}^{\pm}_{\mathbb{F}}$ *refutation of degree* d *from a system of binomials, then there is a* $\mathbf{PC}^{\pm}_{\mathbb{F}}$ *refutation of degree* d *in which each line is a binomial, or a monomial.*

Proof. Recall the linear algebra construction of the basis B_d of the vector space V_d of polynomials derivable in degree d in polynomial calculus proofs.

Even in $\mathbf{PC}^{\pm}_{\mathbb{F}}$, all polynomials are multilinear so the same construction works for $\mathbf{PC}^{\pm}_{\mathbb{F}}$.

This was produced by viewing each polynomial as a vector of coefficients of monomials, iteratively inserting original polynomials as they became available, applying Gaussian elimination and shifting vectors corresponding to multiplication by variables.

Binomials are vectors in which there are only two non-zero entries in this vector notation, one with entry 1, and the other ± 1. Observe that if we start with vectors of this form, shifting a vector will keep it of this form, and any Gaussian elimination at each can keep vectors of this form or simplifis them even further.[2] In particular, Gaussian elimination involving two linearly independent binomial vectors will cancel one zero-coefficient from each using ± 1 coefficients if the result will be a binomial. If the other entries correspond to the same monomial $\prod_{i \in S} z_i$, then they must add up and we can use coefficients $\pm 1/2$ instead of ± 1 in the combination to yield a 1 in that position. Since we know that the input polynomials are unsatisfiable, the Gaussian elimination must eventually end with a vector that corresponds to some single monomial, namely 1; we stop this construction as soon as a single monomial is generated.

[2] Note that this Gaussian elimination is over vectors indexed by monomials of degree at most d in $Z_1, \ldots, z_n$ and not in $x_1, \ldots, x_n$.

The $\mathbf{PC}^{\pm}_{\mathbb{F}}$ derivation simply simulates the vector-based derivation until the monomial $M = \prod_{i \in S} z_i$ is produced. At that point, the $\mathbf{PC}^{\pm}_{\mathbb{F}}$ derivation simply multiplies M by the variables z_i for $i \in S$ one by one in turn, which reduces M to 1 since $z_i^2 = 1$. □

Corollary 3.23. *Let $\mathbb{F}$ have $\mathrm{char}(\mathbb{F}) \neq 2$ and suppose that $F = \{f_1, \ldots, f_m\}$ is a set of binomial polynomials over $z_1, \ldots, z_n$. For every $\mathbf{PC}^{\pm}_{\mathbb{F}}$ refutation of $f_1, \ldots, f_m$ of degree d there is a Gaussian refutation of $\mathscr{L}_F = \ell(f_1), \ldots, \ell(f_m)$ in variables $x_1, \ldots, x_n$ of width at most $2d$. In particular, $\deg(F) \geq w_G(\mathscr{L}_F)/2$.*

Moreover, $\deg(F) \leq \max(w(F), \lceil w_G(\mathscr{L}_F)/2 \rceil)$ and $\mathbf{PC}^{\pm}_{\mathbb{F}}(F)$ is polynomial in n.

Proof. By Theorem 3.22, we can assume without loss of generality that the $\mathbf{PC}^{\pm}_{\mathbb{F}}$ refutation $f_1, \ldots, f_m, \ldots, f_t, \ldots, f_{t+s} = 1$ has each line as a binomial except for lines $f_{t+1}, \ldots, f_{t+s} = 1$, which are monomials. The Gaussian refutation begins with (the simplified forms of) $\ell(f_1), \ldots, \ell(f_t)$. Each $\ell(f_i)$ corresponds to a simplified linear equation of width at most $2d$. Observe that some steps of the polynomial calculus proof such as multiplication by variable z_i yield precisely the same simplified linear equation, but they appear different in the two-sided form. Each $\mathbf{PC}^{\pm}_{\mathbb{F}}$ step deriving a binomial f_j using a ± 1 linear combination of binomials $f_i, f_{i'}$ with simply corresponds to a substitution in equation form and can be derived directly by Gaussian elimination steps of $\ell(f_j)$ from $\ell(f_i)$ and $\ell(f_i)$.

Observe that in the step yielding the monomial f_{t+1}, instead of using the coefficients $\pm 1/2$, if we keep the coefficients ± 1, then we obtain an polynomial $f'_{t+1} = \prod_{i \in S} z_i + \prod_{i \in S} z_i$ which has the translation $\ell(f'_{t+1})$ given

by $\sum_{i\in S} x_i = 1 + \sum_{i\in S} x_i$ whose linear simplified form is $0 = 1$.

The lower bound on $\deg(F)$ immediately follows from the above and Proposition 3.20.

For the other direction, observe that we can simulate any Gaussian inference in $\mathbf{PC}^{\pm}_{\mathbb{F}}$ in a way that nearly exactly balances each binomial so that the two monomials differ in degree by at most 1. □

In particular, the Tseitin formulas for odd-charged graphs have polynomial-size proofs in this basis in which each polynomial is bilinear, and hence represents an equality between parity functions.

Moreover, this representation makes it easy to show that any $\mathbf{PC}^{\pm}_{\mathbb{F}}$ proof of the Tseitin formulas on expander graphs requires large degree.

Theorem 3.24. *For* $\mathrm{char}(\mathbb{F}) \neq 2$*, Tseitin formulas on constant-degree n-vertex* (an, c)*-edge expanders require* $\mathbf{PC}^{\pm}_{\mathbb{F}}$ *(and hence* $\mathbf{PC}_{\mathbb{F}}$*) degree* $\Omega(n)$*).*

To prove this we use the following simple proposition

Proposition 3.25. *Let* $\sum_{i=1}^{k} x_i \equiv b \pmod 2$ *be a parity constraint and let* $C_1, \ldots C_{2^{k-1}}$ *be the set of clauses representing this constraint, then there is a degree k derivation of* $\prod_{i=1}^{k} z_i - (-1)^b$ *from* $p'_{C_1}, \ldots p'_{C_{2^{k-1}}}$ *and vice versa.*

Proof. The single equation form is a semantic consequence of the clausal translation and there are only k variables involved. The same holds in the reverse direction. A Nullstellensatz proof suffices. Details of an explicit proof are an exercise. □

Proof of Theorem 3.24. Let G be an constant degree n-vertex (an, c)-edge expander and ℓ be an odd labeling of G. By the proposition, we can assume without loss of generality that we begin with the binomial form $TS'(G, \ell)$ in the z_i variables of the parity equations for the Tseitin formula $TS(G, \ell)$. By Corollary 3.23, $\deg(TS(G, \ell)) \geq w_G(\mathscr{L}_{TS'(G,\ell)})/2$. By Proposition 3.16, $\mathscr{L}_{TS'(G,\ell)}$ is an (an, c)-boundary expander and hence via Lemma 3.17,

$$w_G(\mathscr{L}_{TS'(G,\ell)}) > can/2.$$

Therefore $\deg(TS(G, \ell)) > can/4$. □

We can immediately use the size-degree relationships of Corollary 3.12 to derive.

Corollary 3.26. *For* $\mathrm{char}(\mathbb{F}) \neq 2$*, Tseitin formulas on constant degree n vertex (an,b)-edge expanders require size* $2^{\Omega(n)}$ $\mathbf{PC}_{\mathbb{F}}$ *and* $\mathbf{PCR}_{\mathbb{F}}$ *proofs.*

Corollary 3.27. *Tseitin formulas on constant degree n vertex (an,b)-edge expanders require size* $2^{\Omega(n)}$ *resolution proofs.*

Theorem 3.28. *For integer* $k > 2$ *and* $\mathrm{char}\mathbb{F}) \neq 2$*, random k-CNF formulas chosen from* $\mathscr{F}_n^{k,m}$ *where* $m = \Delta n$ *for some constant* $\Delta > 0$*, require* $\mathbf{PC}_{\mathbb{F}}$ *degree* $\Omega(n)$ *and size* $2^{\Omega(n)}$ *with probability* $1 - o_n(1)$*.*

Proof. We have already shown that for some constants a and c and such a formula F, the bipartite graph G_F will be an (an, c)-boundary expander with probability $1 - o_n(1)$. However the clauses of F do not correspond to parity equations. In particular, each clause C on a set S of k variables merely asserts that the clause contains at least one positive literal.

The clever idea is that we can add clauses to yield a stronger parity constraint that ensures that the clause is satisfied. In particular, let b be the parity of the unique assignment to the variables in S that makes the C false. Then if we add the remaining $2^{k-1} - 1$ clauses that together with C express that $\sum_{i \in S} x_i = 1 - b \pmod 2$ then we have ruled out any falsifying assignment for C. This yields a new formula F' that is no harder to refute than F since it only has extra clauses added to F. These can be expressed as binomials and the resulting linear constraints $\mathscr{L}_{F'}$ will satisfy $G_{\mathscr{L}_{F'}} = G_F$ and hence will also be an (an, c) boundary expander.

The rest of the argument follows by the same means as the lower bound for Tseitin formulas. □

Theorem 3.29. *Let field $\mathbb{F}$ have* $\mathrm{char}(\mathbb{F}) \neq 2$. $COUNT_n^{2n+1}$ *requires* $\mathbf{PC}_{\mathbb{F}}$ *degree* $\Omega(n)$ *and* $\mathbf{PC}_{\mathbb{F}}$ *or* $\mathbf{PCR}_{\mathbb{F}}$ *proof size* $2^{\Omega(n)}$.

Proof. We can assume without loss of generality that $2n + 1 \equiv 0 \pmod 5$ since $COUNT_{n'}^{2n'+1}$ is a restriction of $COUNT_n^{2n+1}$ for any $n' < n$.

Let $m = (2n + 1)/5$; observe that m is odd. We prove $COUNT_n^{2n+1}$ via reduction[3] from a lower bound for $TS(G, \ell)$ where G is a 4-regular (am, c)-edge expander with an odd number of vertices m and ℓ is the constant function 1. We define a set U of size $2n + 1 = 5m$ as follows: Since it is 4-regular, G has $2m$ edges. For each undirected edge $\{i, j\}$ of G there will be two elements (i, j) and (j, i) in U. There will also be an element i in U for each vertex i of G. This yields a total of $2(2m) + m = 5m = 2n + 1$ elements in U.

Given an assignment to the variables $x_{\{i,j\}}$ of $TS(G, \ell)$, we obtain an assignment to the 2-element subsets of U as follows: if $x_{\{i,j\}} = 1$ then the pair consisting of (i, j) and (j, i) has value 1. Now consider the set U_i consisting of those j such that $x_{\{i,j\}} = 0$. We pair consecutive elements $\{(i, j) \mid j \in U_i\}$ in order of j values. If $|U_i|$ is odd (size 1 or 3), then we pair (i, j) for the largest element $j \in U_i$ with i. All other elements remain unpaired.

Observe that the pairing is always a partial matching and at any vertex i, all elements of U of the form i or (i, j) are paired if and only if U_i is odd which is true if and only if the parity equation in $TS(G, \ell)$ associated with vertex i is satisfied. Thus the totality axioms associated with these elements of U yields the odd-parity constraint in $TS(G, \ell)$.

Observe that the values for the $COUNT_n^{2n+1}$ variables associated with the potential pairing on U can be defined as degree 4 polynomials in the variables of $TS(G, \ell)$ since the graph G has degree 4. We could substitute

[3] One can generalize these lower bounds to proofs of $COUNT_n^{pn+1}$ formulas for prime p in fields of characteristic $q \neq p$ that contain an element $\omega \neq 1$ such that $\omega^p = 1$. One can work with a mod p version of Tseitin formulas that has $2p$ Boolean variables associated with each undirected edge $\{i, j\}$ of G, viewed as two directed edges (i, j) and (j, i), with p variables per direction, expressing that the values in the two directions are in $a_{(i,j)}, a_{(j,i)} \in \{0, 1, \ldots, p-1\}$, and require the constraint $a_{(i,j)} + a_{(j,i)} = p$. A similar handshaking theorem implies that such an assignment of values is impossible. The most natural way to prove this is to work instead with a version of polynomial calculus that involves non-binary assignments to values $1, \omega, \ldots, \omega^{p-1}$ and the equation $x^p - 1 = 0$ instead of $x^2 - 1 = 0$ to prove the degree lower bound and then infer the results for the Boolean version.

these polynomials into a $\mathbf{PC}_{\mathbb{F}}$ refutation of $COUNT_n^{2n+1}$ to yield a refutation of $TS(G,\ell)$ with an increase of degree by at most a factor of 4. Therefore the $\mathbf{PC}_{\mathbb{F}}$ degree of $COUNT_n^{2n+1}$ is at least 1/4 of the $\mathbf{PC}_{\mathbb{F}}$ degree of $TS(G,\ell)$. □

Finally we note that the following holds. The proof is an exercise.

Proposition 3.30. *Let* $\mathrm{char}(\mathbb{F}) \neq 2$. *Any unsatisfiable set of binomials F in variables $z_1,\ldots,z_n$ has a polynomial-size* $\mathbf{PC}_{\mathbb{F}}^{\pm}$ *refutation.*

3.5 Lower bound for the Pigeonhole Principle in Polynomial Calculus

Theorem 3.31. *For any field* $\mathbb{F}$, PHP_n^m *requires* $\mathbf{PC}_{\mathbb{F}}$ *degree* $\lceil n/2 \rceil$.

Proof. The proof applies to $function\text{-}PHP_n^m$ which implies the same result for PHP_n^m. The proof of this theorem follows by explicit computation of a basis B_d for the vector space V_d of multilinear polynomials of degree $\leq d$ that are generated by the $function\text{-}PHP_n^m$ axioms. The Hole and Function axioms set any monomial $x_S = \prod_{ij\in S} x_{ij} = 0$ that does not correspond to a partial matching of pigeons to holes. We assume that we work modulo these non-matching polynomials as well as modulo the ideal I.

Observe that this also means that any $x_{ij}\cdot Q_i$ is 0 because all terms of Q_i that involve some $x_{ij'}$ for $j' \neq j$ are 0 because they violate a Hole clause and the remaining terms yield $x_{ij}^2 - x_{ij}$ which is 0 in the ideal I. Therefore, any polynomial of degree at most d modulo this ideal that could possibly be generated by the $function\text{-}PHP_n^m$ constraints (using possibly larger degree intermediate polynomials) is in the vector space T_d generated by all polynomials of the form $x_M \cdot \prod_{i\in A} Q_i$ for some partial matching M and subset of pigeons A where M and A involve disjoint sets of pigeons.

Let $\mathscr{M}_d$ be the set of all such partial matchings of size at most d. Given a partial matching, we call matched holes *occupied*. The only axioms we have not yet handled are the pigeon axioms. In order to describe the basis B_d of V_d, we need to identify a special subset of $\mathscr{M}_d$. For this we define an operation on elements of $\mathscr{M}_d$ that has been called the *pigeon dance*.

Definition 3.32. Let $M = \{(i_1,j_1),\ldots,(i_t,j_t)\} \in \mathscr{M}_d$. A *pigeon dance* on M repeats the following steps:

- For $k = 1$ to t
  - If all holes $> j_k$ are currently occupied then fail.
  - Otherwise, replace (i_k,j_k) with some (i_k,j_k') where hole $j_k' > j_k$ is not currently occupied.

If successful, let $M = M_0, M_1, \ldots M_t$ be the sequence of matchings after each step.

A pigeon dance is *minimal* iff at each step j'_k is the *smallest* unoccupied hole $> j_k$. If this does not fail then it is unique and we define $D(M)$ to be the resulting matching M_t in the minimal dance if it exists.

The pigeon dance is not deterministic and it may not even be possible to complete a pigeon dance.

Proposition 3.33. *Some pigeon dance on M succeeds iff $D(M)$ is defined.*

Proof. Intuitively, the high numbered holes are the precious resource for the pigeon dance and a minimal pigeon dance leaves them freer. More formally, one can show how to convert the steps of any successful pigeon dance to a minimal one by swapping: Suppose that the first $s-1$ steps of a pigeon dance are minimal. If the s-the step is not then there is some unoccupied hole $j > j_s$ in M_{s-1} such that $j'_s > j > j_s$. We have two cases:

If j is unoccupied in M_t then we simply replace (i_s, j'_s) by (i_s, j) in all matchings $M_s, \ldots, M_t$, yielding a pigeon dance with $\geq s$ minimal steps.

If j is occupied in M_t then there is some step $s' > s$ such that pigeon $i_{s'}$ is mapped to $j > j_{s'}$. Observing that this also implies $j'_s > j_{s'}$, we replace $(i_{,s}, j'_s)$ by (i_s, j) in all matchings $M_s, \ldots, M_{s'-1}$ and we replace $\{(i_s, j'_s), (i_{s'}, j)\}$ by $\{(i_s, j), (i_{s'}, j'_s)\}$ in all matchings $M_{s'}, \ldots, M_t$, again yielding a pigeon dance that is minimal for $\geq s$ steps.

Applying this for all $s \leq t$ yields the minimal dance. □

Proposition 3.34. *If $D(M) = D(M')$ are defined then $M = M'$.*

Proof. Observe that at each step s of a minimal pigeon dance, we can reconstruct M_{s-1} from M_s by replacing (i_s, j'_s) by (i_s, j) where j is the largest unoccupied hole less than j'_s in M_s. □

The basis B_d is given by the set of all polynomials of the form $x_M \cdot \prod_{i \in A} Q_i$ where:

- M is a matching on $[m] \times [n]$ for which $D(M)$ is defined.
- $\emptyset \neq A \subset [m]$ is a set of pigeons that is disjoint from the pigeons matched in M.
- $|M| + |A| \leq d$.

Proposition 3.35. *If $d \leq \lceil n/2 \rceil$ then B_d is a basis for V_d.*

Proof Sketch. One can show that if we add $B'_d = \{x_M \mid D(M)$is defined and $|M| \leq d\}$ to B_d and $d \leq \lceil n/2 \rceil$ then together they form a basis for T_d. The key to the argument that this is a basis is to use a partial ordering on pairs (M, A) and argue by induction that any generator for T_d that has a failed pigeon dance in M is spanned by terms in $B_d \cup B'_d$. In particular, $(M, A) \prec (M', A')$ if $M \cup A \subsetneq M' \cup A'$ or $M \cup A = M' \cup A'$ and the largest $i \in M \cup A$ is matched

to a hole j' in M' where either $i \in A$ or i is matched to a lower hole j than j' in M.

The idea of the argument is to prove by induction over the partial order that for any (M',A'), a term of the form $x_M \cdot \prod_{i\in A} Q_i$ is equivalent modulo the basis B_d to the sum of all possible terms based on (M',A') such that either (1) $A' = A$ and M' is the result of a single pigeon dance step on M or (2) $A' = A - \{i\}$ and $M = M' \cup (i,j)$.

This is sufficient since we can take a single pair (M,A) where $D(M)$ does not exist and apply these steps until the pigeon dance fails. At the point the set of terms equivalent to the (M,A) terms is empty. □

This is enough since the polynomial $1 \in B'_d$ (corresponding to $M = \emptyset$) and since $B_d \cup B'_d$ is a basis of T_d its representation in that basis is unique, so 1 cannot be generated by B_d. □

Theorem 3.36. *$fPHP^m_n$ requires $\mathbf{PC}_{\mathbb{F}}$ and $\mathbf{PCR}_{\mathbb{F}}$ size $2^{\Omega(n)}$.*

Proof Sketch. Even though we have am $\Omega(n)$ degree lower bound for refutations of $fPHP^m_n$, the fact that it has mn variables means that the generic conversion from small size to low degree does not yield a size lower bound.

However, we can instead make a small modification of the general argument that is tailored especially to the $fPHP^m_n$ formulas. In this variant of the conversion from small size to small degree, we call a monomial big iff it mentions at least D holes.

Rather than choosing a most frequently occurring variable or literal to set, we choose a most frequently mentioned hole among the at most S big monomials. By averaging, some hole j must be mentioned in at least SD/n big monomials. We now simply choose any variable x_{ij} associated with hole j in these monomials, and apply one of two restrictions: We either set $x_{ij} = 0$, or we set $x_{ij} = 1$ and all $x_{i'j} = 0$ for $i' \neq i$ (the latter is forced by the hole constraints).

In the first case, any monomial containing x_{ij} is set to 0 and in the second case, any monomial mentioning hole j that does not contain x_{ij} is set to 0. At least one of these two cases removes at least $DS/(2n)$ of the big monomials mentioning hole j, leaving at most $(1 - D/(2n)) \cdot S$ monomials remaining[4]. In the other case the number of variables has been reduced so we can apply induction as before. This yields the same parameters as before but with the number of holes n replacing the number of variables.

Pluggling the degree bound into this new relationship yields the lower bound. □

[4] Since the monomials of interest correspond to partial matchings, each monomial has at most one variable mentioning each hole. Because of this, if we choose the least frequently occurring x_{ij} then the case when x_{ij} is set to 1 will remove almost all monomials mentioning hole j, which can do somewhat better than $(1 - S/(2n))$

3.6 Lower bounds for $\mathbf{PC}^{\pm}_{\mathbb{F}}$ proof size using diameter

The small refutation of, for example, Tseitin formulas shows that there is no such size-degree theorem for $\mathbf{PC}^{\pm}_{\mathbb{F}}$. In particular, restrictions that set logical

variables to 0 or 1 and hence remove entire monomials in the $\{0,1\}$ representation of $\mathbf{PC}_{\mathbb{F}}$ merely shorten monomials in the $\{1,-1\}$ representations of $\mathbf{PC}^{\pm}_{\mathbb{F}}$.

Nonetheless, there is another measure of $\mathbf{PC}^{\pm}_{\mathbb{F}}$ proofs, that is implicit in the work of Sokolov that does allow us to apply a similar restriction argument.

Definition 3.37. For $S \subseteq [n]$, we use the notation $z_S = \prod_{i \in S} z_i$, and for a polynomial p, we write $\mathrm{Mon}(p)$ for the set of S such that z_S occurs with non-zero coefficient in p. The *diameter* of a multilinear polynomial $p \in \mathbb{F}[z_1, \ldots, z_n]$, $\mathrm{diam}(p)$, is the maximum size of the symmetric difference between the support of any pair of monomials z_S in p, i.e. $\mathrm{diam}(p) = \max_{S,T \in \mathrm{Mon}(p)} |S \oplus T|$. The diameter of a $\mathbf{PC}^{\pm}_{\mathbb{F}}$ proof is the maximum diameter of any polynomial in the proof.

Observe that the diameter of a multilinear polynomial is the diameter of its Newton polytope in the ℓ_1 metric. Note also that for a clause C, the diameter of p'_C, as well as its degree, is precisely the length of C.

Definition 3.38. For a multilinear polynomial p, define the set $[p]$ of multilinear polynomials equivalent to $z_S \cdot p$ modulo $I_{\pm 1}$ for some monomial z_S of p.

Proposition 3.39. *If $q = z_S \cdot p$ for polynomials p and q, there is a $\mathbf{PC}^{\pm}_{\mathbb{F}}$ derivation of q from p of degree at most* $(\deg(p) + \deg(q) + |S|)/2$.

Proof. We obtain q from p by multiplying p by the variables in z_S one by one and reducing at each step as needed. Observe that in each step the degree changes by at most 1. Let k be the maximum degree attained at any intermediate step; it takes at least $k - \deg(p)$ steps to reach this degree. From there, the degree must decrease to $\deg(q)$, which requires at least a further $k - \deg(q)$ steps. The total number of steps is $|S|$. Therefore $k - \deg(p) + k - \deg(q) \leq |S|$. and hence $2k \leq \deg(p) + \deg(q) + |S|$. □

Proposition 3.40. *If $q \in [p]$ then*

(a) $\deg(q) \leq \mathrm{diam}(p)$*,*

(b) $\mathrm{diam}(q) = \mathrm{diam}(p)$*,*

(c) $[q] = [p]$*, and*

(d) there is a $\mathbf{PC}^{\pm}_{\mathbb{F}}$ derivation of q from p of degree at most $\deg(p) + \mathrm{diam}(p)/2$.

Proof. Let $q = z_S \cdot p$ modulo $I_{\pm 1}$ for some $S \in \mathrm{Mon}(p)$.

(a): For $S, T \subseteq [n]$, $z_S \cdot z_T = z_{S \oplus T}$ modulo $I_{\pm 1}$. Therefore, since $S \in \mathrm{Mon}(p)$, the degree of q is at most $\max_{T \in \mathrm{Mon}(p)} |S \oplus T| \leq \max_{S,T \in \mathrm{Mon}(p)} |S \oplus T| = \mathrm{diam}(p)$.

(b): $\mathrm{diam}(q) = \max_{T,T' \in \mathrm{Mon}(p)} |(S \oplus T) \oplus (S \oplus T')| = \max_{T,T' \in \mathrm{Mon}(p)} |T \oplus T'| = \mathrm{diam}(p)$

(c): Let $q' \in [q]$, so $q' = z_{S'} \cdot q$ modulo $I_{\pm 1}$ for some $S' \in \mathrm{Mon}(q)$; i.e. $S' = S \oplus T$ for some $T \in \mathrm{Mon}(p)$. Therefore, $S' \oplus S = T$ and so $q' = z_{S'} \cdot z_S \cdot p = z_T \cdot p$ modulo $I_{\pm 1}$ and hence $q' \in [p]$.

Conversely, let $q' \in [p]$. Then $q' = z_{T'} \cdot p$ modulo $I_{\pm 1}$ for some $T' \in \mathrm{Mon}(p)$. Then by definition $S \oplus T' \in \mathrm{Mon}(q)$ and $q' = z_{S \oplus T'} \cdot z_S \cdot p = z_{S \oplus T'} \cdot q$ modulo $I_{\pm 1}$ and hence $q' \in [q]$.

(d): Let $q \in [p]$. By definition, $q = z_S \cdot p$ modulo $I_{\pm 1}$ for some $S \in \mathrm{Mon}(p)$. The bound follows by plugging $|S| \leq \deg(p)$ into Proposition 3.39. □

The following example shows that the degree bound given in part (d) is the best possible.

Example 3.41. Let ℓ and k be positive integers with $2k \neq \ell$. Let T be an index set of size $\ell + k$ and define $p = \sum_{S \subset T, |S| = \ell} z_S$. Then $\deg(p) = \ell$ and $\mathrm{diam}(p) = 2k$.

Any $\mathbf{PC}^{\pm}_{\mathbb{F}}$ derivation of any $q \in [p]$ from p must multiply p by a subset of the variables indexed by T, one by one. If this goes on for k steps, the resulting polynomial will contain the term z_T and hence has degree $k + \ell$, so it remains to prove that k steps are needed.

If $2k < \ell$, then $\deg(q) \leq \mathrm{diam}(p) < \deg(p)$, but in the first k steps of any derivation the degree can only increase, so $> k$ steps are required.

In general, the argument requires a little more work. Observe that for any $q \in [p]$, the degree $2k$ portion of q is $\sum_{B \subseteq T \setminus A,\ |B| = k} z_{A \cup B}$ for some $A \subset T$ with $|A| = k$. Let C be the set of indices of the first $i < k$ variables that are used to multiply p in producing q. Observe that the resulting portion of maximum degree $\ell + i$ is $\sum_{D \subset T \setminus C,\ |D| = \ell} z_{C \cup D}$. In particular, the $i < k$ common indices in these maximum degree terms cannot possibly match the k common indices in the maximum degree terms in q.

The following lemma implies that a low diameter $\mathbf{PC}^{\pm}_{\mathbb{F}}$ refutation can be converted to a low degree refutation with only a small increase in size. (For that conclusion one would only need the proof for some sequence of polynomials, rather than for every sequence.)

Lemma 3.42. *Let d_0 be the maximum degree of $f_1, \ldots, f_m$. If there is a $\mathbf{PC}^{\pm}_{\mathbb{F}}$ proof from $f_1, \ldots, f_m$, $(f_1, \ldots, f_T)$ of diameter at most $d \geq 1$ and size at most s, then, for any sequence of polynomials $f'_1, \ldots, f'_T$ with each $f'_i \in [f_i]$, there is a $\mathbf{PC}^{\pm}_{\mathbb{F}}$ proof $(f_1, \ldots, f_m, \ldots, f'_1, \ldots, f'_T)$ from $f_1, \ldots, f_m$ of degree at most $\max(d, d_0) + d/2$ and size $O((d + d_0)s)$.*

Proof. We show how to derive each f'_i in the required degree and size at $O(d + d_0)$ times the size to derive f_i:

1. If f_i is an axiom then by definition it has degree at most d_0. Therefore $f'_i \in [f_i]$ is derivable in degree at most $d_0 + d/2$ by Proposition 3.40. This takes at most d_0 steps and each line in that derivation has exactly the same number of monomials as f_i.

2. If $f_i = z_k \cdot f_j$ where $j < i$, then we observe that $[f_i] = [f_j]$. Therefore $f_i' = z_R \cdot f_j$ for some $R \in \mathrm{Mon}(f_j)$. Now $f_j' = z_S \cdot f_j$ for some $S \in \mathrm{Mon}(f_j)$ and hence
$$f_i' = z_R \cdot f_j = z_{R \oplus S} \cdot z_S \cdot f_j = z_{R \oplus S} \cdot f_j'.$$
Since $\mathrm{diam}(f_j) \leq d$, we have $|R \oplus S| \leq d$ and hence, by Proposition 3.39, f_i' is derivable in degree at most $3d/2$ in at most d steps from f_j'; each line of this derivation has the same number of monomials as f_i does.

3. If $f_i = a \cdot f_j + b \cdot f_{j'}$ where $j, j' < i$. Then $f_i' = z_R \cdot f_i$ for some $R \in \mathrm{Mon}(f_i)$, $f_j' = z_S \cdot f_j$ for some $S \in \mathrm{Mon}(f_j)$ and $f_{j'}' = z_T \cdot f_{j'}$ for some $T \in \mathrm{Mon}(f_{j'})$. We have two cases:

   $\mathrm{Mon}(f_j)$ and $\mathrm{Mon}(f_{j'})$ are disjoint: Therefore $\mathrm{Mon}(f_i)$ is their union, and so $S, T \in \mathrm{Mon}(f_i)$. Since f_i has diameter at most d, $|R \oplus S| \leq d$ and $|R \oplus T| \leq d$. Now
   $$f_i' = z_R \cdot f_i = a \cdot z_{R \oplus S} \cdot z_S \cdot f_j + b \cdot z_{R \oplus T} \cdot z_T \cdot f_{j'} = a \cdot z_{R \oplus S} \cdot f_j' + b \cdot z_{R \oplus T} \cdot f_{j'}'.$$
   Since there is no cancellation of monomials, every monomial in $z_{R \oplus S} \cdot f_j'$ and every monomial in $z_{R \oplus T} \cdot f_{j'}'$ is a monomial in f and hence each has degree at most d. Therefore, since $\deg(z_{R \oplus S})$, $\deg(z_{R \oplus T}) \leq d$, by applying 3.39 twice, we see that we derive f_i' from f_j' and $f_{j'}'$ via a further degree $3d/2$ inference in at most $2d + 1$ lines, each line of which has a number of monomials equal to the number in f_i or f_j or $f_{j'}$,

   There is some $U \in \mathrm{Mon}(f_j) \cap \mathrm{Mon}(f_{j'})$: From f_j' and $f_{j'}'$, we first derive $p = z_{U \oplus S} \cdot f_j' = z_U \cdot f_j$ and $q = z_{U \oplus T} \cdot f_{j'}' = z_U \cdot f_{j'}$, each of which has degree at most d since $\mathrm{diam}(f_j), \mathrm{diam}(f_{j'}) \leq d$ and can be derived via a proof of degree $3d/2$ of at most d lines. Hence
   $$r = a \cdot p + b \cdot q = z_U \cdot (a \cdot f_j + b \cdot f_{j'}) = z_U \cdot f_i$$
   has degree at most d. Assume without loss of generality that $R \in \mathrm{Mon}(f_j)$. Since the diameter of f_j is at most d, $|R \oplus U| \leq d$. Therefore $f = z_R \cdot f_i = z_{R \oplus U} \cdot z_U \cdot f_i = z_{R \oplus U} \cdot r$ can be derived from r in degree at most $3d/2$ by Proposition 3.39 since $|R \oplus U| \leq d$, $\deg(r) \leq d$ and $\deg(f) \leq d$. There are at most $3d + 1$ lines in total for this derivation, each of which has size bounded by the sizes of f_i, f_j, $f_{j'}$.

□

Definition 3.43. Given $\mathbf{PC}_{\mathbb{F}}^{\pm}$ proof $\Pi = (f_1, \ldots, f_T)$ and a positive integer D, we write
$$W(\Pi, D) = \{A \subseteq [n] \mid A = R \oplus S \text{ for some } R, S \in \mathrm{Mon}(f_t) \text{ with } f_t \in \Pi \text{ and } |A| > D\}$$
for the set of *wide* symmetric differences in Π.

Obviously, if $W(\Pi, D) = \varnothing$ then Π has diameter at most D. Sokolov [5] developed a way to iteratively reduce the set of wide symmetric differences in a way similar to that of Clegg, Edmonds, and Impagliazzo [6], and Impagliazzo, Pudlak, and Sgall [7]. Unlike those results, this method will not allow us to convert small size $\mathbf{PC}^{\pm}_{\mathbb{F}}$ proofs into small diameter proofs based on the original set of constraints; such a conversion is impossible as shown by the same Tseitin formula examples. However, it will let us produce small diameter derivations involving restrictions of the original set of constraints.

Definition 3.44. For any set of constraints $F = \{f_1 = 0, \ldots, f_m = 0\}$ on $\{x_1, \ldots, x_n\}$ and set of indices $B \subseteq [n]$, write F_B for the subset of F consisting of all polynomial constraints in F that contain some variable indexed by B.

The following theorem is the key diameter-reducing result. It differs from the degree reduction over $\{0,1\}$ in that the variables being set are *not* allowed to be among the ones that contribute to the large diameter. It is similar in the way that one finds a small set of variables that is contained in (and can be used to remove) all but a small number of wide contributions.

Theorem 3.45. *Let $D > 0$ and let $\mathbb{F}$ be a field with* $\mathrm{char}(\mathbb{F}) \neq 2$. *Let $F = \{f_1 = 0, \ldots, f_m = 0\}$ be a set of polynomial constraints on variables $x_1, \ldots, x_n$ and Π be a $\mathbf{PC}^{\pm}_{\mathbb{F}}$ refutation of F. Then for every integer $k \geq 1$ there is a subset $B \subseteq [n]$ with $|B| \leq k$ such that for every restriction ρ defined on a subset of $[n] \setminus B$ that satisfies F_B, there is a $\mathbf{PC}^{\pm}_{\mathbb{F}}$ refutation Π' of $F|_\rho$ with $|W(\Pi', D)| \leq (1 - D/n)^k \cdot |W(\Pi, D)|$.*

Proof. We choose $B \subseteq [n]$ to be a set of size k that intersects the largest number of sets in $W(\Pi, D)$.

In particular, we can choose B inductively as follows: At each step we choose the index $i \in [n]$ that appears most frequently among the sets in $W(\Pi, D)$ that do not contain previously chosen indices in $[n]$. By averaging, since every set in $W(\Pi, D)$ has size $> D$, at each stage there is some index i in $[n]$ that is in at least a D/n fraction of previously untouched sets. Therefore, the number of sets in $W(\Pi, D)$ that are disjoint from B is at most $(1 - D/n)^k \cdot |W(\Pi, D)|$.

By soundness, applying the restriction ρ to the $\mathbf{PC}^{\pm}_{\mathbb{F}}$ refutation Π of F yields a refutation $\Pi|_\rho$ of $F|_\rho$. Every element of $W(\Pi|_\rho, D)$ corresponds to a unique element of $W(\Pi, D)$ shortened by the removal of indices set by ρ. Since ρ does not set any value indexed by B, the number of elements in $W(\Pi|_\rho, D)$ that are disjoint from B is at most the number in $W(\Pi, D)$ that are disjoint from B. Therefore, in order to prove the theorem, it will suffice to show that we can modify $\Pi|_\rho$ to get rid of all elements of $W(\Pi|_\rho, D)$ that intersect B.

Observe that since ρ satisfies F_B, $F|_\rho$ does not contain any constraints with variables indexed by B. (The satisfied constraints in F_B have been reduced to 0 and can be ignored.) Since ρ does not set any variables in B,

$\Pi|_\rho$ will still have terms containing the variables in B. We show that these variables in B merely serve a kind of book-keeping role and can be removed:

For each $f_t|_\rho$ in $\Pi|_\rho$, we write $f_t|_\rho = \sum_{A\subseteq B} z_A \cdot p_{t,A}$ where $p_{t,A}$ is a polynomial that does not include any variable whose index is in B. The new refutation Π' replaces each polynomial $f_t|_\rho$ in $\Pi|_\rho$ by the sequence of all $p_{t,A}$ that are non-zero. To see that this is still a $\mathbf{PC}^{\pm}_{\mathbb{F}}$ refutation of $F|_\rho$, we observe that

- $f_j|_\rho = p_{j,\emptyset}$ and $p_{j,A} = 0$ for $\emptyset \neq A \subseteq B$ for all $j \leq m$ (the axioms),
- if $f_j|_\rho = z_i \cdot f_{j'}|_\rho$ for $i \notin B$ then $p_{j,A} = z_i \cdot p_{j',A}$ for every $A \subseteq B$,
- If $f_j|_\rho = z_i \cdot f_{j'}|_\rho$ for $i \in B$ then $p_{j,A} = p_{j',A\oplus\{i\}}$ for every $A \subseteq B$,
- If $f_j|_\rho = a \cdot f_{j'}|_\rho + b \cdot f_{j''}|_\rho$ then $p_{j,A} = a \cdot p_{j',A} + b \cdot p_{j'',A}$ for every $A \subseteq B$, and
- $p_{T,\emptyset} = 1$

We finally observe that the symmetric differences of monomials in Π' are precisely those of $\Pi|_\rho$ that do not contain any variable indexed by B, which is at most the number for Π and hence $|W(\Pi', D)| \leq (1 - D/n)^k \cdot |W(\Pi, D)|$ as required. □

In order for this theorem to be useful, it should be the case that the input constraints F_B that contain the variables indexed by B can be satisfied without making the restriction ρ too large.

Remark 3.46. As in the case of Lemma 3.42, one can extend Theorem 3.45 to other non-zero values for true, a, and false, b, with essentially no loss. The only difference is in the definition of $p_{j,A}$ when $f_j|_\rho = z_i \cdot f_{j'}|_\rho$ for $i \in B$: Instead of $p_{j',A\oplus\{i\}}$, if $i \in A$, we have $p_{j,A} = (a+b)\cdot p_{j',A} + p_{j',A\setminus\{i\}}$ and if $i \notin A$, we have $p_{j,A} = -ab \cdot p_{j,A\cup\{i\}}$.

4
Communication and Interpolation

In this chapter we focus on two related techniques for understanding proof complexity, inspired by the problem of falsified clause search problem Search_F associated with CNF formula F that we previously discussed in the context of resolution proof complexity in Section 2.6. Both are related to *communication complexity*.

4.1 Basics of Communication Complexity

In communication complexity, players each receive information about a subset of the input variables associated with a function or relation f, which they must compute on their joint inputs by communicating over a broadcast channel (or blackboard). The *communication complexity* of f is the minimum of the worst-case number of bits they need to communicate in order to produce a correct answer. The constraints are purely information-theoretic in that players are assumed to have unbounded computational power. Unless otherwise specified there are only two parties, also known as players, and often referred to as Alice and Bob, who received disjoint portions of the input, which is viewed as a pair $(x, y) \in X \times Y$. The two-party communication complexity of a function/relation f on $X \times Y$ is denoted by $C(f)$. One can observe that after each bit of communication, the set of inputs consistent with the communication so far is a set of the form $A \times B$ where $A \subseteq X$ and $B \subseteq Y$, and object that is known as a combinatorial *rectangle*.

Sometimes players are also allowed to make random choices and have a probability at most ε of producing an incorrect answer; the communication complexity in this case is denoted by $C_\varepsilon(f)$. There is also a variant of randomized communication complexity in which players begin by sharing an arbitrarily long uniformly random string and otherwise must behave deterministically. This can reduce the number of bits to be sent by at most an $O(\log n)$ additive amount for n-bit input and can be nicer to reason about. This complexity is denoted by $C_\varepsilon^{pub}(f)$.

Example 4.1 (Equality)**.** Consider the *equality* predicate $EQ_n : X \times Y \to$

$\{0,1\}$ where $X = Y = \{0,1\}^n$ and $EQ_n(x,y) = 1$ iff $x = y$. We have $C(EQ_n) = n+1$. The upper bound, which holds for all Boolean functions on this domain comes from the trivial algorithm in which Alice sends her entire input to Bob who responds with the answer. The lower bound can be shown by observing that the only combinatorial rectangles in $X \times Y$ on which the answer is 1 are the 2^n disjoint rectangles consisting of the single points (x,x) for $x \in \{0,1\}^n$. This means that there must be $> 2^n$ possible markings on the blackboard and hence $> n$ bits communicated.

On the other hand, $C_\varepsilon^{pub}(EQ_n)$ can be seen to be at most $O(\log(1/\varepsilon)$ by the following algorithm: Alice and Bob interpret the shared random string as a sequence of $k = \lceil \log_2(1/\varepsilon) \rceil$ strings $r_1, \ldots, r_k \in \{0,1\}^n$. Alice computes $b_i = r_i^T \cdot x \bmod 2$ for $i \in [k]$ and sends $b_1, \ldots, b_k$ to Bob who checks that $b_i = r_i^T \cdot y \bmod 2$ for all $i \in [k]$.

Example 4.2 (Disjointness)**.** Consider the *set disjointness*[1] predicate $DISJ_n : X \times Y \to \{0,1\}$ where $X = Y = \{0,1\}^n$ and $DISJ_n(x,y) = 1$ iff there exists an $i \in [n]$ such that $x_i = y_i = 1$. One can show that $C(DISJ_n) = n+1$ by arguing that no two pairs of the form (x,x) can be in the same combinatorial rectangle.

[1] The name is based on viewing elements of $\{0,1\}^n$ as characteristic vectors of subsets of $[n]$. $DISJ_n(x,y) = 0$ iff the corresponding sets are disjoint. It might have been more natural to use the term "set intersection" but the term"disjointness" was originally used.

Unlike the savings using randomness for EQ_n, for any fixed $\varepsilon < 1/2$, $C_\varepsilon^{pub}(DISJ_n)$ is $\Omega(n)$. This requires a quite non-trivial proof. This lower bound for randomized algorithms for set disjointness has been the basis of many of the most important applications of communication complexity.

When there are more than $k > 2$ players, there are two generalizations of two-party communication complexity typically considered: one called the *number-in-hand (NIH)* model in which the inputs are partitioned between the k players, and the other called the *number-on-forehead (NOF)* model in which each input is not seen by precisely one of the k players, as in typical logic puzzles in which players try to make deductions based on marks on their foreheads. Both are described by partitions of the inputs into k blocks, with the i-th block consisting of those inputs seen by player i in the NIH model, and those inputs *not* seen by player i in the NOF model. Write the complexities as $C^{k,NOF}(f)$, $C^{k,NIH}(f)$ and $C_\varepsilon^{k,NOF}(f)$, $C_\varepsilon^{k,NIH}$ for the case of k players when the partition of the inputs is fixed. The randomized multiparty communication complexity of the computation of *polynomial threshold functions*:

Example 4.3 (Linear Inequalities)**.** Define $GT_{n,k}$ to be the function that takes k signed integers $y_1, \ldots, y_k \in [-2^n, 2^n]$ and outputs 1 iff $y_1 + \cdots + y_k \geq 0$. Then trivially $C^{k,NIH}(GT_{n,k})$ is $O(nk)$, but one can show that for any $c \geq 0$, $C_{1/n^c}^{k,NIH}(GT_{n,k})$ is $O(k \log^2 n)$.

Example 4.4 (k-Party Disjointness)**.** The function $DISJ_n^k : (\{0,1\}^n)^k \to \{0,1\}$ has value 1 on input $x^1, \ldots, x^k \in \{0,1\}^n$ iff there is an $i \in [n]$ such that $x_i^1 = \cdots = x_i^k = 1$.

It is known that for any fixed $\varepsilon > 0$, $C_\varepsilon^{k,NOF}(DISJ_n^k)$ is at least $\sqrt{n}/(k2^k)$. This lower bound is actually proven by showing a stronger result that the same lower bound also holds when we are given a promise that there is at most one $i \in [n]$ with $x_i^1 = \cdots = x_i^k = 1$, a partial function denoted by $UDISJ_n^k$ for unique disjointness.

4.2 Tree-like proofs and the communication complexity of Search$_F$

Lemma 4.5. *Let* **P** *be a dynamic proof system with lines that for n-variable inputs express Boolean functions from some class $B_n^{\mathbf{P}}$ and inference rules that derive each line from at most two predecessors. Suppose further that under every partition of the n input variables into k pieces, every function $f \in B_n^{\mathbf{P}}$ can be evaluated by a k-party (randomized) communication complexity protocol of complexity at most $C(n)$ (and error at most $\varepsilon_n > 0$).*

Then, if CNF formula F has a tree-like refutation in **P** *of size at most S, then Search$_F$ can be evaluated by a k-party protocol of the same type with communication complexity $O(C(n)\log S)$ (and error at most $O(\varepsilon \cdot \log S)$ under every partition of the inputs into k pieces.*

Proof. Suppose that there is tree-like refutation R of F of size at most S. Fix a partition of $[n]$ into k pieces. The players will use R to guide their search.

The key property we use is that in any binary tree T with S leaves, there is some node v that has between $1/3$ and $2/3$ of the leaves as descendants. This follows by the usual argument following backwards from the root of T taking the child with more descendants until that number is at most $2/3$. (The players don't need to communicate to know this node.)

The communication protocol for Search$_F$ on input $x \in \{0,1\}^n$ proceeds as follows: The players use the protocol for $B_n^{\mathbf{P}}$ to evaluate the line f at node v of R. If f evaluates to false, then the search reduces to a recursive computation on the subtree T_v^0 in R rooted at v. If f evaluates to true, then the search reduces to a recursive computation in the tree T_v^1 which consists of R with the subtree T_v^0 removed.

Correctness follows from the soundness of the proof system. Since both T_V^0 and T_v^1 have at most he number of levels of recursion is $O(\log S)$ which yields both complexity bounds. □

Since any clause with inputs divided between two players can be evaluated by a 2-bit protocol we have:

Corollary 4.6. *If F has a tree resolution refutation of size at most S, then $C(Search_F)$ is $O(\log S)$ under any partition of the inputs.*

Corollary 4.7. *If CNF formula F in n variables has a tree-like* **CP*** *refutation of at most S line then under any partition of the inputs, $C(Search_F)$ is $O(\log n \cdot \log S)$. Moreover, if F has a tree-like* **CP** *protocol of at most S lines then for any fixed $\varepsilon > 0$, $C_\varepsilon(Search_F)$ is $O((\log n)^2 \log S)$.*

Proof. Observe that each line in $\mathbf{CP}^*$ is of the form $a_1\ x_1 + \cdots + a_n\ x_n - b \geq 0$ where each a_i and b is at most $n^{O(1)}$ in absolute value and hence under any partition of the inputs, they can compute the sum of the terms they can see (say, including the $-b$ with Bob) to get y_1 for Alice and y_2 for Bob. Observe that y_1 and y_2 are also of absolute value at most $n^{O(1)}$. It now remains for them to evaluate $GT_{N,2}(y_1, y_2)$ where N is $O(\log n)$. Plugging in the trivial protocol for $GT_{N,2}$ yields the bound.

For the case of **CP**, by Proposition 1.26 we can assume that each coefficient in a line is at most $2^{(n \log_2 n)/2-1}$ and since there are $n+1$ of them, the corresponding y_1 and y_2 have absolute value at most 2^N where N is $O(n \log n)$. Since, without loss of generality, $\log S$ is at most n, and the error per step can be taken to be $1/n^2$, we get that the total error is $o(1)$. □

Corollary 4.8. *Let $\mathbf{P}_d$ be a dynamic propositional proof system whose lines are polynomial inequalities of degree at most d. If CNF formula F in n variables has a tree-like $\mathbf{P}_d$ refutation of with at most S line then for any $\varepsilon > 0$, $C_\varepsilon^{d+1,NOF}(Search_F)$ is at most $O(d^4(\log n)^2 \log S)$ under any partition of the inputs into d pieces.*

If we are further guaranteed that every coefficient in the tree-like proof has absolute value at most $n^{O(d)}$, then $C^{d+1,NOF}(Search_F)$ is at most $O(d^2 \log n \cdot \log S)$.

Proof. The basic idea is that since the degree of any line $p \geq 0$ of a $\mathbf{P}_d$ proof is at most d, every monomial in p will contain variables on the foreheads of at most d players, so there is one player left over who can see all the values. If we attribute each monomial to the smallest numbered player who can see all of its variables. Each player i first computes the weighted sum y_i of all the monomials in p attributed to it. Then the players combine to compute $GT_{N,d+1}$ on their y_i values using the NIH protocol. It remains to understand how large N needs to be. There are only $\binom{n}{\leq d}$ which is $n' = n^{O(d)}$ monomials and each takes on a value in $\{0,1\}$, so we can apply Proposition 1.26 to say that each coefficient only needs to be $O(n' \log n')$ bits long. Therefore each y_i is only $N = O(n' \log n')$ bits long. Since $\log N$ is $O(d \log n)$, we obtain the claimed result using the bound from Example 4.3.

If we know *a priori* that the coefficients are at most $n^{O(d)}$ in value, then any of the y_i is also at most $n^{O(d)}$ in absolute value or $N = O(d \log n)$ bits long. □

Definition 4.9 (k-fold Tseitin)**.** Given a graph G, an odd labelling ℓ of the vertices V of G, and an integer $k \geq 2$, we define the *k-fold Tseitin* formulas to be CNF formulas $TS(G,\ell)^{(\wedge k)}$, in which we replace each edge variable of the Tseitin formula by a logical AND of new variables and expand the result using the distributive law to a CNF formula. Observe that if G has n vertices and maximum degree Δ then $TS(G,\ell)^{(\wedge k)}$ has size at most $N = (2k)^\Delta n$.

For a certain family of graphs G, the k-fold Tseitin formulas on G can be

shown to be hard examples for any dynamic tree-like proof system based on polynomial inequalities. The graph G_n on n vertices in the original proof is the union of an explicitly constructed highly expanding $\Delta = \Theta(\log n)$-regular graph and a Hamiltonian path.

Theorem 4.10. *Let $k \geq 2$ and $m = n^{1/3}/\log n$. If ℓ is an odd labeling of G_n then any tree-like $\mathbf{P}_{k-1}$ refutation of $TS(G_n, \ell)^{(\wedge k)}$ requires size at least $2^{\Omega(C_{(\varepsilon}^{k,NOF}(UDISJ_m^k)/\log n)^{1/3}}$.*

Proof Sketch. By applying Corollary 4.8, it suffices to obtain a lower bound on the k-party NOF communication complexity of $\text{Search}_{TS(G_n,\ell)^{(\wedge k)}}$. The only thing that we need to show is a randomized reduction to this communication problem from $UDISJ_m^k$ under a suitable partition of the variables.

That partition allocates each of the k players with one of the k variables associated to each edge of the k-fold Tseitin formula. Now solving $\text{Search}_{TS(G_n,\ell)^{(\wedge k)}}$ finds a charge violation given an odd labeling. By a standard randomized reduction, finding a charge violation for an odd labeling of G_n can also be used to see whether an even labeling has no violations[2].

Each of the m positions in the disjointness problem is associated with edge-disjoint paths between m pairs of vertices in G_n (that do not use the edges of the Hamiltonian path part of G_n). The input to the disjointness problem is transformed in such a way that the result has even parity everywhere except at a pair of endpoint vertices associated with an intersecting coordinate. Therefore charge violations of the all 0's charge occur if and only if the sets intersect. □

[2] Given an even labeling, flip the label of a random vertex and see if the odd-charge violation returned is the this planted one. If not, we know that there was a violation of the even labeling. By careful choice, one arrange it so that any violations of the odd-labelling search problem are essentially indistinguishable from each other, so there is no advantage in guessing which was an original violation and which was planted.

Combining the above with the lower bounds for NOF k-party unique set disjointness yields the following:

Corollary 4.11. *There is a $\delta > 0$ such that for $2 \leq k \leq \delta \log n$, and any odd labeling ℓ of G_n, degree $k-1$ tree-like Positivstellensatz Calculus refutations of $TS(G_n, \ell)^{(\wedge k)}$ require size $2^{n^{\Omega(1)}}$ which is $2^{N^{\Omega(1/\log k)}}$ where N is the size of $TS(G_n, \ell)^{(\wedge k)}$.*

4.3 Feasible Interpolation

A classic theorem in logic is Craig's interpolation theorem. If there are formulas ϕ and ψ that have shared variables X but also have other variables. It says that if $\phi \to \psi$, then one can "interpolate" a formula θ only involving variables X, between ϕ and ψ so that $\phi \to \theta \to \psi$.

There is a natural version of interpolation involving CNF formulas where one can easily prove this statement. Let $A(x, y)$ and $B(x, z)$ be two CNF formulas and suppose that $F(x, y, z) = A(x, y) \wedge B(x, z)$ is unsatisfiable. Therefore there can't be any pair (y, z) so that both $A(x, y)$ and $B(x, z)$ are

satisfiable, so the following (partial) function makes sense:

$$C(x) = \begin{cases} 1 & \text{if } \exists y.\, A(x,y) \\ 0 & \text{if } \exists z.\, B(x,z) \\ * & \text{otherwise.} \end{cases}$$

Any Boolean function that is consistent with C is called an *interpolant*[3] for $A(x,y) \wedge B(x,z)$.

[3] Observe that $\neg(A(x,y) \wedge B(x,z)) \equiv \neg A(x,y) \vee \neg B(x,z) \equiv A(x,y) \rightarrow \neg B(x,z)$ and that $A(x,y) \rightarrow C(x)$ and $C(x) \rightarrow \neg B(x,z)$.

We will show that there is a direct relationship between certain proof systems and interpolants. The general pattern will be that if a CNF formula F of the above type has an efficient refutation in some proof system, then it has an interpolant that can be computed efficiently. Sch proof systems will be said to have *feasible interpolation*.

Example 4.12 (Clique-Coloring Formulas)**.** These formulas, denoted $CLIQUE{-}COLOR_{n,k}$, express the (impossible) claim that n-vertex graph with a k-clique can be $k-1$ colorable. They have variables

- x_e for each potential edge $e \in \binom{[n]}{2}$ in an n-vertex graph G.
- y_{ij} for each $i \in [k]$ and $j \in [n]$ expressing that vertex j is the i-th vertex in a chosen k-clique in G.
- z_{jc} for each $j \in [n]$ and $c \in [k-1]$ expressing that vertex j has color c in a chosen $(k-1)$-coloring of G.

We write $CLIQUE{-}COLOR_{n,k} = A(x,y) \wedge B(x,z)$ where $A(x,y)$ expresses that y is k-clique in G and is given by the clauses:

- $y_{i1} \vee \cdots \vee y_{in}$ for each $i \in [k]$,
- $\neg y_{ij} \vee \neg y_{i'j'} \vee x_{\{j,j'\}}$ for each $i \neq i' \in [k]$ and $j \neq j' \in [n]$,
- $\neg y_{ij} \vee \neg y_{ij'}$ for each $i \in [k]$ and $j \neq j' \in [n]$,

and $B(x,z)$ expresses that z is a $(k-1)$-coloring of G and is given by the clauses:

- $z_{j1} \vee \cdots \vee z_{j(k-1)}$ for each $j \in [n]$,
- $\neg x_{\{j,j'\}} \vee \neg z_{jc} \vee \neg z_{j'c}$ for each $j \neq j' \in [n]$ and each $c \in [k-1]$,
- $\neg z_{ic} \vee \neg z_{ic'}$ for each $i \in [n]$ and $c \neq c' \in [k-1]$.

The unsatisfiability of $CLIQUE{-}COLOR_{n,k}$ is really an indirect property of the pigeonhole principle PHP^k_{k-1}; The constraints ensure that y and z express functions whose composition would be a 1-1 function from $[k]$ to $[k-1]$. Like that principle, it is still unsatisfiable if the third clause types in both $A(x,y)$ and $B(y,x)$ are removed, since these merely ensure that y and z express functions rather than relations, whose composition would also violate PHP^k_{k-1}.

Observe that any interpolant C for the $CLIQUE{-}COLOR_{n,k}$ must be able to distinguish between graphs with k-cliques and those that are $(k-1)$-colorable, which is an NP-hard problem.

Observe also that the interpolant C can be taken to be a *monotone* function of x (that is, flipping bits of x from 0 to 1 can only cause the value of C from 0 to 1), since adding edges to a graph can only increase the set of possible k-cliques and reduce the set of possible $(k-1)$-colorings. This property of having a monotone interpolant is not special to $CLIQUE{-}COLOR_{n,k}$ and definition holds for any instance for which the x variables occur only positively in $A(x,y)$.

We say that a circuit computing a monotone function $C(x)$ is itself monotone if it has $\wedge$ and $\vee$ gates but no $\neg$ gates. It is known that any monotone circuit that separates k-cliques from $(k-1)$-colorings for $k = O((n/\log n)^{2/3})$ requires exponential size $2^{\Omega(k^{1/2})}$.

Theorem 4.13. *Let $F(x,y,z) = A(x,y) \wedge B(x,z)$ be an unsatisfiable CNF formula. If F has a resolution refutation of size at most S then there is an interpolant for F computable by a circuit of size at most $4S$.*

Furthermore, if x only occurs positively in $A(x,y)$, then that circuit can be monotone.

Proof. The structure of the circuit for the interpolant for F will follow the DAG structure of a minimal resolution refutation of F. We assign each leaf of the proof labeled by a clause in $A(x,y)$ by 0 and each leaf of the proof labeled by a clause in $B(x,z)$ by 1. We will assign each internal node that resolves on some y_j by an $\vee$-gate and each one that resolves on some z'_j by an $\wedge$-gate.

Each input x_i will only enter in the circuit at the nodes in the proof that resolve on variable x_i. Suppose that clause $C' \vee D'$ at vertex u is the resolvent of clauses $C = C' \vee x_i$ and $D = D' \vee \neg x_i$ labeling children v and w respectively. Then we assign a gate $\mathrm{sel}(x_i, v, w)$ to node v, where $\mathrm{sel}(0, v, w)$ gives the value of node v and $\mathrm{sel})(1, v, w)$ gives the value of node w. We can write $\mathrm{sel}(x_i, v, w) = (x_i \wedge w) \vee (\neg x_i \wedge v)$ which is only 4 gates, so the size bound is as claimed.

It remains to argue that this circuit computes an interpolant for F. If we take this refutation and apply the restriction $x \leftarrow a_x$ to it, then each time we have a vertex v that resolves one of the x_i variables, one of its child clauses is set to 1 by the assignment a_x, so we view it as a copying operation for the other clause, resulting in a clause that is a subset of the original one labeling v. Observe that, with this translation, any connection between clauses derived from $A(x,y)$ and those derived from $B(x,z)$ is broken and only one of those sets of clauses leads to the root. (All other resolution steps for which both contain the resolution variable on y_i both have children among those derived from $A(x,y)$ and for steps on z_i, both are derived from $B(x,z)$.)

However, there may be resolution steps that no longer make sense because one or both of the child clauses no longer contains the resolution variable; in that case we simply pass one of those clauses to the parent, with the provisos that (1) if the resolution variable is a y variable and one of the child clauses is derived from $B(x,z)$ then we choose to pass that one through, and (2) if that variables is a z variable and one of the child clauses is derived from $A(x,y)$ then we pass that one through. The resulting proof is either a refutation of $A(x,y)|_{x\leftarrow a_x}$ or of $B(x,z)|_{x\leftarrow a_x}$.

We claim by induction that the function computed at each node in our circuit on input a_x, has value 0 if it is derived from $A(x,y)|_{x\leftarrow a_x}$ and has value 1 if it is derived from $B(x,z)|_{x\leftarrow a_x}$. This is sufficient for correctness of the circuit, since the output node will be have value 0 if there is a refutation of $A(x,y)|_{x\leftarrow a_x}$ (which is ensured if $B(x,z)|_{x\leftarrow a_x}$ is satisfiable) and will have value 1 if there is a refutation of $B(x,z)|_{x\leftarrow a_x}$ (which is ensured if $A(x,y)|_{x\leftarrow a_x}$ is satisfiable).

Clearly, the claim is true at the leaves by construction. If it is true inductively, then the sel gate will correctly choose the derivation type for any node resolving on some x_i. Finally, our choice to pass through the clause of opposite type to the resolution variable at a useless resolution step ensures that the labeling of gates resolving on y_j by $\vee$ and $z_{j'}$ by $\wedge$ ensures that the type of the derived clause is correctly represented in the circuit.

Observe that if x only occurs positively in $A(x,y)$, then in the above construction, whenever we resolve on some x_i, any clause derived solely from $A(x,y)$ contains x_i positively so we can replace the function $\mathrm{sel}(x_i,v,w)$ by the monotone function $(x_i \vee v) \wedge w$, which differs from $\mathrm{sel}(x_i,v,w)$ on precisely one assignment ($x_i = 0$; $v = 1$; $w = 0$) where it evaluates to 0, because if a child is derived from A then we can always assume that it is the one passed through if $x_i = 0$. □

Corollary 4.14. *$CLIQUE{-}COLOR_{k,n}$ requires resolution refutations of size $2^{\Omega(k^{1/2})}$ for k that is $O((n/\log n)^{2/3})$.*

While this is very nice, we have many other lower bounds for resolution, so it may not be surprising. On the other hand there is a more general version of monotone version of interpolation that applies to cutting planes proofs, which yield the first lower bounds for general cutting planes. The difference is that we require a much more general and powerful notion of monotone circuit.

Definition 4.15. A k-ary *monotone real function* is a mapping $f : \mathbb{R}^k \to \mathbb{R}$ with that property that increasing the value of any of its arguments cannot decrease the value of the function.

A *monotone real circuit* is a directed acyclic graph of in-degree at most 2 with each node of indegree $k = 1, 2$ labeled by a k-ary monotone real function. It computes a monotone real function of its inputs as a composition of the functions labeling its gates.

The following theorem due to Pudlak is the key to strong lower bounds for **CP**.

Theorem 4.16. *Let $F(x,y,z) = A(x,y) \wedge B(x,z)$ be an unsatisfiable CNF formula such that x only occurs positively in $A(x,y)$. If F has a* **CP** *refutation of size at most S then there is an interpolant for F computable by a monotone real circuit of size at most $S + |F| \cdot w(F)$.*

Proof. The basic structure of the argument is similar to the proof of Theorem 4.13. However, given an assignment $x = \alpha$, instead of thinking of whether to keep the part of a line that depends on $A(x,y)$ or on $B(x,z)$, we will keep both and figure things out afterward.

More precisely, given an assignment $x = \alpha$, we will split each line

$$a(x) + b(y) + c(z) \geq D$$

of a **CP** refutation into two lines

$$b(y) \geq D_0 \qquad \text{and} \qquad c(z) \geq D_1$$

where integers D_0 and D_1 satisfy $D_0 + D_1 \geq D - a(\alpha)$. The part involving y will be derived from $A(x,y)|_{x \leftarrow \alpha}$ and the part involving z will be derivated from $B(x,z)|_{x \leftarrow \alpha}$.

In particular, this will split the last line of the proof, $0 \geq 1$, into $0 \geq D_0$ and $0 \geq D_1$ such that $D_0 + D_1 \geq 1$. Since D_0 and D_1 are integers at least one of them will be ≥ 1 and hence one of the two sides of the split will be a refutation of $A(x,y)$ or $B(x,z)$ under the restriction $x \leftarrow \alpha$ using precisely the same rules as each line of the original proof.

We produce this split by induction. An axiom $a(x) + b(y) \geq D$ derived from $A(x,y)$ yields a split under restriction of the form

$$b(y) \geq D - a(\alpha) \qquad \text{and} \qquad 0 \geq 0.$$

An axiom from $B(x,z)$ of the form $a(x) + c(z) \geq D$ becomes

$$c(z) \geq D - a(\alpha) \qquad \text{and} \qquad 0 \geq 0.$$

We now argue each inference rule in turn. For multiplication by a $K > 0$, we just multiply each side of the split by K. For addition, suppose that we are summing $a(x)+b(y)+c(z) \geq D$ and $a'(x)+b'(y)+c'(z) \geq D'$ to obtain $a(x) + a'(x) + b(y) + b'(y) + c(z) + c'(x) \geq D + D'$. Then by the inductive hypothesis we have the splits

$$b(y) \geq D_0 \qquad \text{and} \qquad c(z) \geq D_1$$

and

$$b'(y) \geq D'_0 \qquad \text{and} \qquad c'(z) \geq D'_1$$

where $D_0 + D_1 \geq D - a(\alpha)$ and $D'_0 + D'_1 \geq D' - a'(\alpha)$. The split for the sum can be chosen as:

$$b(y) + b'(y) \geq D_0 + D'_0 \qquad \text{and} \qquad c(z) + c'(z) \geq D_1 + D'_1.$$

Note that $D_0 + D'_0 + D_1 + D'_1 \geq D + D' - a(\alpha) - a'(\alpha)$ as required.

Finally, suppose that we apply the division rule: Given $K \cdot a(x) + K \cdot b(y) + K \cdot c(z) \geq D$ with $K > 0$ we derive $a(x) + b(y) + c(z) \geq \lceil D/K \rceil$. By induction we have a split $K \cdot b(y) \geq D_0$ and $K \cdot c(z) \geq D_1$ where $D_0 + D_1 \geq D - K \cdot a(\alpha)$.

Since $b(y)$ and $c(z)$ take on integer values, by applying the same division rule to each of the split inequalities we have $b(y) \geq \lceil D_0/K \rceil$ and $c(z) \geq \lceil D_1/K \rceil$. Now

$$\begin{aligned} \lceil D_0/K \rceil + \lceil D_1/K \rceil &\geq \lceil (D_0 + D_1)/K \rceil \\ &\geq \lceil (D - K \cdot a(\alpha))/K \rceil \qquad \text{by assumption} \\ &= \lceil D/K \rceil - a(alpha) \end{aligned}$$

which means that this new split is also correct.

The algorithm that will be given by the real monotone circuit of fan-in at most 2 will simply compute the (negation of the) D_0 quantity for the split version of each line as a function of its input α. For the initial inequalities, this is a quantity of the form $a(\alpha) - D$ which is a monotone function in α since each variable x_i appears positively in the original clause. This requires at most $w(F)$ fan-in 2 monotone real gates for each clause in F. For each inference rule we have a single gate that is either (i) multiplication by a positive constant, (ii) addition, or (iii) division by positive constant and rounding down (since it is computing the negative a number that is being rounded up), each of which is a monotone function of one or two arguments. At the end, when it has the final constraint $0 \geq D_0$, it will output 0 if $D_0 \geq 1$ (which means that the left side of the splits is a refutation of $A(x, y)$) and output 1 otherwise. This is monotone in $-D_0$. □

A version of the same theorem holds for the non-monotone case, but the non-monotone version of these circuits seems extremely powerful. Since the variables only are Boolean, and the original constraints have integer coefficients, we can assume without loss of generality that the proofs have integer values of at most $O(n \log n)$ bits each, and hence the resulting circuits only need to deal with moderately sized integer values rather than arbitrary reals.

Nonetheless, reasoning about such circuits in terms of arbitrary reals seems to held avoid distractions. In particular, Pudlak showed that the monotone Boolean circuit lower bound separating graphs with k-cliques from those that are $(k-1)$-colorable applies equally well to monotone Boolean circuits.

Corollary 4.17. *$CLIQUE{-}COLOR_{k,n}$ requires* **CP** *refutations of size $2^{\Omega(k^{1/2})}$ for any k that is $O((n/\log n)^{2/3})$.*

This was the very first exponential lower bound for **CP** and for many years was (essentially) a singular example of a hard problem for **CP** because of the specialized format of the formulas required and the difficulty proving monotone (real) circuit lower bounds.

However, recently, two new methods have been developed to obtain lower bounds for **CP**, one of which is based on *lifting* with we will describe in a separate chapter, and the other is a re-interpretation of interpolation that allows one to apply it to formulas with arbitrary format.

The general idea of the second method is the following: Given a CNF formula $F = \wedge_{i\in[m]} C_i$ with variables from X, we show that we can prove **CP** lower bounds for F by proving interpolation lower bounds for many different formulas based on F, each indexed by a partition of X into two parts, X_0 and X_1, which we denote by F^{X_0,X_1} that will also have extra variables.

In F^{X_0,X_1}, the variables in X_0 and X_1 play the role of the z and y variables in the monotone interpolation framework. F^{X_0,X_1} will also have a new set of m variables, W, which will play the role of the x variables in the interpolation framework. In F^{X_0,X_1} we replace each clause C_i by two clauses:

$$(C_i^1 \vee w_i) \wedge (\neg w_i \vee C_i^0)$$

where C_i^1 is the sub-clause of C_i on the variables of X_1 and C_i^0 is the sub-clause of C_i on the variables of X_0. We can write $A(x_1, w)$ for the conjunction of all the $(C_i^1 \vee w_i)$ clauses and $B(x_0, w)$ for all the $(\neg w_i \vee C_i^0)$ clauses. Clearly w only appears positively in $A(x_1, w)$ so any interpolant for F^{X_0,X_1} is monotone. Moreover, each F^{X_0,X_1} is logically equivalent for F.

Proposition 4.18. *For any CNF formula F with m clauses on variables X and any partition $X = X_0 \cup X_1$, if F has a* **CP** *(resolution) refutation of size at most S then F^{X_0,X_1} has a* **CP** *(respectively, resolution) refutation of size at most $S + m$*

Proof. Simply apply one resolution step to eliminate the variables in W or the equivalent step in **CP**. □

Therefore, in order to provide a lower bound for F, it suffices to prove a monotone (real) circuit complexity lower bound for F^{X_0,X_1} for the interpolant for just one partition (X_0, X_1).

What are the requirements for such an interpolant? There is precisely one variable in W for each clause of F. A truth assignment to W corresponds to a selection of a subset of clauses $T \subseteq [m]$ so we can view the interpolant as a function f defined on subsets of $[m]$ with the following correctness property[4]:

- If $\wedge_{i\notin T} C_i^1$ is satisfiable then $F(T) = 1$, and
- if $\wedge_{i\in T} C_i^0$ is satisfiable then $F(T) = 0$.

[4] Observe that such an interpolant can be monotone in T since increasing the size of T makes it easier it is to satisfy the first constraint, and force the function value to 1, which reducing T makes it easier to force te function value to 0 in the second constraint

This is a nice enough definition in its own right that we give it an independent name and call it an *unsatisfiability certificate* for F under the partition (X_0, X_1).

Monotone interpolation immediately implies the following[5]:

[5] In the case of resolution there actually is a direct construction of the monotone circuit for the unsatisfiability certificate for F. In the resolution refutation we replace each input clause C_i by a variable y_i, each resolution step on a variable in X_0 by $\wedge$, and each resolution step on a variable in X_1 by $\vee$. The proof of correctness is left as an exercise.

Proposition 4.19. *If F on variables X has a resolution (**CP**) refutation of size S then for every partition of X into $X_0 \cup X_1$, F has a and unsatisfiability certificate f under (X_0, X_1) computable by a monotone (real) circuit of size $O(S)$ (respectively, $O(S + m \cdot w(F))$.*

We omit the proof of the following theorem, whose proof is quite nontrivial.

Theorem 4.20. *There is a constant $c > 0$ such that if X is a set of $2n$ variables, $k \geq c \log n$ and F is a random k-CNF formula in $m = O(2^k n)$ clauses on X, then for a fixed partition of X into two sets X_0 and X_1 of size n, any unsatisfiability certificate for F under partition (X_0, X_1) requires monotone real circuit size $2^{n^{\Omega(1)}}$ with probability $1 - o(1)$.*

From this, we immediately obtain.

Corollary 4.21. *For some constants $c, c' > 0$, with probability $1 - o(1)$, random k=CNF formulas for $k = \lceil c \log n \rceil$ with $O(n^{c'})$ clauses are unsatisfiable but require **CP** refutations of size $2^{n^{\Omega(1)}}$.*

Similar results also hold for a concise version of the pigeonhole principle known as the *bit pigeonhole principle*, $BPHP_n^m$, which is defined only when n is a power of 2 and $m > n$; write $n = 2^\ell$ Instead of the usual $m \times n$ matrix of variables, $BPHP_n^m$ has $m \times \ell = m \times \log_2 n$ variables. The variables in the i-th row are the binary encoding of the name of the hole that the i-th pigeon maps to. The clauses simply state that no two rows of the matrix are equal. That is, we have $n \cdot \binom{m}{2}$ clauses of length 2ℓ:

- $\bigvee_{j \in [\ell]-1} (\neg x_{ij}^{b_j} \vee \neg x_{i'j}^{b_j})$ for each $i \neq i' \in [m]$ and each $b_1, \ldots b_\ell \in \{0,1\}$,

where we use the notation $x^1 = x$ and $x^0 = \neg x$.

Both of these classes of hard examples have clauses of logarithmic length. This seems fundamental in order to be able to apply this technique since finding good partitions of variables for constant-sized clauses seems hard.

Open Problem 4.22. *Prove lower bounds on **CP** refutation size for unsatisfiable random k-CNF formulas for constant k (or $k = o(\log n)$).*