Circuits, Proofs and Propositional Model Counting

Sravanthi Chede
Indian Institute of Technology Ropar, Rupnagar, India

Leroy Chew
TU Wien, Austria

Anil Shukla
Indian Institute of Technology Ropar, Rupnagar, India

Abstract

In this paper we present a new proof system framework CLIP (Circuit Linear Induction Proposition) for propositional model counting (#SAT). A CLIP proof firstly involves a Boolean circuit, calculating the cumulative function (or running count) of models counted up to a point, and secondly a propositional proof arguing for the correctness of the circuit.

This concept is remarkably simple and CLIP is modular so it allows us to use existing checking formats from propositional logic, especially strong proof systems. CLIP has polynomial-size proofs for XOR-pairs which are known to require exponential-size proofs in MICE [16]. The existence of a strong proof system that can tackle these hard problems was posed as an open problem in Beyersdorff et al. [3]. In addition, CLIP systems can p-simulate all other existing #SAT proofs systems (KCPS(#SAT) [8], CPOG [4], MICE). Furthermore, CLIP has a theoretical advantage over the other #SAT proof systems in the sense that CLIP only has lower bounds from its propositional proof system or if $\mathsf{P}^{\#\mathsf{P}}$ is not contained in P/poly, which is a major open problem in circuit complexity.

CLIP uses unrestricted circuits in its proof as compared to restricted structures used by the existing #SAT proof systems. In this way, CLIP avoids hardness or limitations due to circuit restrictions.

2012 ACM Subject Classification Theory of computation → Proof complexity

Keywords and phrases Propositional model counting, Boolean circuits, #SAT, Proof Systems, Certified Partition Operation Graph (CPOG)

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2024.18

Funding *Leroy Chew*: This work was supported by FWF ESPRIT grant no. ESP 197.

Acknowledgements We thank Olaf Beyersdorff, Tim Hoffmann, Luc Nicolas Spachmann for useful discussions on this topic. We also thank the anonymous reviewers for helpful suggestions on the paper.

1 Introduction

Given a propositional formula, the problem of finding its total number of satisfying assignments (models) is known as the propositional model counting problem #SAT [24]. The problem is known to be #P-complete and is considered one of the hardest problem in the field of computational complexity. In fact, it is known that with a single call to a #SAT-oracle, any problem from polynomial hierarchy can be solved in polynomial time (Toda's Theorem [27]).

Over the last few years, some important proof systems have been developed for #SAT. The knowledge compilation based proof system (KCPS(#SAT)) [8] is the first non-trivial proof system designed for #SAT. A KCPS(#SAT) proof for a CNF represents the proposition as a decision-DNNF (Decomposable Negation Normal Form), with some additional annotations for checking. A decision-DNNF allows for model counting to be easily extracted. However, limitations and lower bounds for KCPS(#SAT) have already been established [2, 8].

The second proof system designed for #SAT is MICE (Model-counting Induction by Claim Extension) [16]. Unlike KCPS(#SAT), it is a line based proof system which computes the model count in a step-by-step fashion using some simple inference rules. Several lower bounds for MICE have been established in the literature [2, 3]. For example, XOR-PAIRS [3], are shown to be hard for the MICE proof system. Recently, an important proof system CPOG [4] was introduced for #SAT. Similar to KCPS(#SAT), CPOG is also based on knowledge compilation. A CPOG proof for a CNF formula ϕ consists of a Partitioned-Operation Graph (POG) G along with a Resolution proof of the fact that $G \equiv \phi$.

The relationship between these three proof systems are now well known. It has been shown in [2], that CPOG is exponentially stronger than KCPS(#SAT) and MICE. On the other hand, KCPS(#SAT) and MICE are incomparable [2, Figure 1]. This means that KCPS(#SAT) and MICE have unconditional lower bounds. For CPOG a lower bound is currently unknown, but its proof complexity is necessarily tied to the limitations of POGs.

In this paper, we introduce a new #SAT proof format CLIP (Circuit Linear Induction Proposition) (Definition 5). A CLIP based proof for a CNF formula ϕ consists of a Boolean circuit calculating the running count of models up to an assignment treated in some fixed lexicographical order. We denote this Boolean circuit as a cumulator (Definition 3). In addition to the cumulator, the CLIP proof also contains a certificate proving the correctness of the cumulator. The CLIP format is similar to CPOG in the sense that instead of a POG, CLIP has a cumulator. Since POG uses restricted versions of AND and OR gates, as compared to cumulators, we believe that CLIP format is much stronger than CPOG. In this direction, we show that CLIP can p-simulate CPOG (Theorem 30). In fact, we show that CLIP has lower bounds only if some major open problems of proof complexity or circuit complexity are solved (Theorem 7).

In MICE and KCPS(#SAT), proofs can grow exponentially because of unsatisfiable formulas that have lower bounds in Resolution. In this direction, for any unsatisfiable formula which is hard in Resolution, it is unclear how large the CPOG proofs will be. However, for unsatisfiable formulas in CLIP, proofs are no bigger than their shortest DRAT proofs (Proposition 35). In addition, we also show that XOR-PAIRS which are known to be hard for existing proof systems are easy for the CLIP format (Theorem 34). We sum up our contributions in the Figure 1. We explain the same in detail in the following subsection.

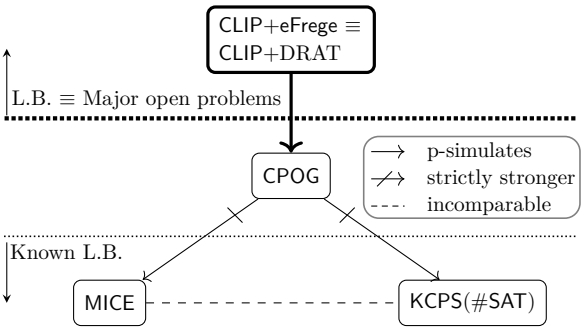

Figure 1 *Hierarchy of #SAT proof systems. New results are shown in bold.*

Our Contributions

1. **Introducing a new proof system framework for #SAT (CLIP)**: We present a proof system where proofs are pairs containing a circuit and a propositional proof. The circuit is a multi-output Boolean circuit we call the *cumulator* which takes a complete assignment α and returns the number (in binary) of models of a propositional formula up to α in some fixed lexicographical ordering of assignments. In addition, CLIP proofs also contain a certificate showing the correctness of the cumulator. The certificate here is a propositional proof. This is possible because we can construct a tautology that covers every inductive step for any two consecutive complete assignments. The CLIP format allows us to use any known propositional proof system $\mathcal{P}$ for proving the correctness of the cumulator. In this paper we focus on CLIP+ Extended Frege (CLIP+eFrege). We show that CLIP+eFrege is a powerful proof system in the sense that it has a lower bound only if a super-polynomial lower bound for eFrege is found or it is proven that $\mathsf{P}^{\#\mathsf{P}} \not\subseteq \mathsf{P/poly}$ (Theorem 7). Hence proving lower bounds in CLIP+eFrege will lead to solving major open problems in the fields of proof complexity or circuit complexity. Another way to say this is that CLIP is the first #SAT proof system which is conditionally optimal. Note that such systems already exist in the propositional and QBF worlds, i.e. IPS (Ideal Proof System) [18] and eFrege +$\forall$red [1] system respectively.
2. **A CLIP+eFrege simulation technique for all existing #SAT proof systems**: The CLIP+eFrege simulation technique consists of three parts. The first part consists of extracting a cumulator from any #SAT proof system which is closed under restrictions. The second part establishes that if a #SAT proof system admits easy eFrege proofs of the "properties of restriction" (Definition 15) then it can be p-simulated by CLIP+eFrege. The final part proves that the CPOG system admits all the required properties and hence can be p-simulated by CLIP+eFrege. Let us briefly explain each of them separately.
   a. **Cumulator extraction** (Section 4.1). We show that from any #SAT proof system which is closed under restrictions (Definition 1), there is a simple technique to efficiently extract a cumulator from its proofs. For this, we carefully use the concept of Fenwick trees [15, 26] to introduce and compute the Fenwick assignments (Definition 11). Informally, the Fenwick assignments are a small set of partial assignments that collectively covers all assignments up to a given complete assignment (Definition 9).
   b. **Extended Frege (eFrege) certification of the cumulator** (Section 4.2). Informally, the properties of restriction imply that after restricting a proof with a complete assignment α, the model count will be "one" or "Zero" depending on whether α satisfies the formula or not. Whereas, in the case of restricting a proof with a partial assignment α undefined on some variable x, the model count returned should be the sum of model counts returned when restricting seperately with α_0 and α_1, where $\alpha_b = \alpha \cup \{x = b\}$. These two are the only properties we need to know which when globally combined tell us that the model count under the restriction of a partial assignment is correct. If a #SAT proof system admits easy eFrege proofs for these properties of restriction (Definition 15), we show that it can be p-simulated by CLIP+eFrege (Theorem 19).
   c. **CLIP+eFrege p-simulates CPOG** (Section 5). Using the structure of the POG to form an inductive proof. We explicitly show that the CPOG proof system admits easy eFrege proofs of the properties of restriction (Lemma 29). Thereby, proving that CLIP+eFrege p-simulates CPOG (Theorem 30), which in-turn p-simulates KCPS(#SAT) and MICE [2]. This shows that CLIP+eFrege p-simulates all existing #SAT proof systems.

3. **Upperbounds in CLIP for some hard formulas of existing #SAT systems**:
   a. **XOR-PAIRS**. Since the CLIP framework uses unrestricted Boolean circuits in its proof as compared to other existing #SAT proof systems, CLIP is capable of handling formulas that are hard for other systems. We show this for the family XOR-PAIRS, which are known to be hard for MICE [3], and give an easy proof for the same in the CLIP+eFrege proof system (Theorem 34). For the short proof, we first carefully define a short cumulator for the XOR-PAIRS. Then, using a constant case analysis we certify the correctness of the cumulator in eFrege.
   b. **Unsatisfiable formulas**. We show that any unsatisfiable formula which has an easy eFrege proof, also has an easy CLIP+eFrege proof (Proposition 35). It is already known that for unsatisfiable formulas, MICE and KCPS(#SAT) are p-equivalent to Resolution and regular-Resolution respectively [2, Proposition 5.1, 5.3]. As a result, all unsatisfiable formulas, which are hard for Resolution and easy for eFrege are all hard for MICE and KCPS(#SAT) but easy for CLIP+eFrege. We list three such important counting based unsatisfiable formulas. Namely, the pigeonhole principle (PHP), the clique-coloring principle [22, Definition 7.1] and the Random Parity principle, which are known to be hard for Resolution [19, 22, 9] but are easy for eFrege [13, 6, 7, 10].

2 Preliminaries

For a Boolean variable x, its literals can be x or $\neg x$. We use the notation $\overline{\ell} = \neg x$ when $\ell = x$ and $\overline{\ell} = x$ when $\ell = \neg x$. A clause C is a disjunction of literals and a conjunctive normal form (CNF) formula ϕ is a conjunction of clauses. We denote the empty clause by $\bot$. $vars(\phi)$ is the set of all variables in formula ϕ.

2.1 Assignments

A partial assignment is a partial mapping from a set of propositional variables X to $\{0,1\}$, when the mapping is defined everywhere we say the assignment is complete. $\langle X \rangle$ is the set of all complete assignments. Consider a totally ordered set of variables X. An **initial assignment** α is a partial assignment to X such that there are no pairs $x, y \in X$ where $x < y$ and x is undefined in α and y is defined. $vars(\alpha)$ are the variables for which α is defined and $|\alpha|$ represents $|vars(\alpha)|$. A partial assignment α can be extended to a total assignment by appending 0/1 assignment to the variables $X \setminus vars(\alpha)$. Two partial assignments α, β are called non-overlapping, if there does not exist any total assignment γ which can be obtained by extending both α and β.

For a CNF ϕ, $\phi|_\alpha$ (similarly $C|_\alpha$) denotes the restricted formula (or clause) resulting from replacing all occurrences of $vars(\alpha)$ in ϕ (or C) with assignments from α. For a propositional formula F we define the indicator function $\mathbb{1}_F$, this acts on the free variables of F. $\mathbb{1}_F$ is equal to an assignment that corresponds to $\mathbf{0}$ when F is false and $\mathbf{1}$ when F is true.

When variables are ordered as $X =< x_{n-1}, \ldots x_0 >$, complete assignments can be seen as binary numbers i.e. $\{x_2 = 1, x_1 = 0, x_0 = 1\}$ represents $\mathbf{5}$.

Let num map assignments to integers using the standard binary encoding ($num(\alpha) = \sum_{i=0}^{i<n} \mathbb{1}_{\alpha(x_i)} \cdot 2^i$), and num^{-1} be its inverse. We also encapsulate arithmetic statements with $||\cdot||$ to indicate that we revert this into a proposition. Later we will drop this notation when obvious. We denote $[J]$ to denote numbers $\{\mathbf{1}, \mathbf{2}, \ldots, J-\mathbf{1}, J\}$ and $[J_1, J_2]$ to denote numbers $\{J_1, J_1+\mathbf{1}, \ldots, J_2-\mathbf{1}, J_2\}$. We distinguish numerals $\mathbf{0}$ and $\mathbf{1}$ from Boolean constants 0 and 1 through the use of boldface.

Given a formula ϕ over a set of variables X, a model is an complete assignment to X that satisfies ϕ. The set of models of ϕ is $\mathcal{M}(\phi)$. We denote the total number of models of a CNF ϕ as $\#_{\text{models}}(\phi) = |\mathcal{M}(\phi)|$. #SAT is the computational problem of calculating $\#_{\text{models}}(\phi)$ from a CNF ϕ. The class of languages decidable in polynomial time with an oracle to #SAT are denoted by $P^{\#P}$.

2.2 Circuits

A Boolean *circuit* σ on variables X is a directed acyclic graph, in which the input nodes (with in-degree 0) are Boolean variables $\in X$ and other nodes are the basic Boolean operations: $\vee$ (OR), $\wedge$ (AND) and $\neg$ (NOT) and have in-degree at most 2. Every Boolean circuit σ evaluates a Boolean function whose output is that of the node with out-degree 0 in σ. P/poly is the class of Boolean functions computed by polynomial-sized circuit families.

We refer to a *multi-circuit* when we have multiple nodes with out-degree 0. This is simultaneously many overlapping circuits. Multi-circuits take inputs and outputs of Boolean vectors of fixed length. We denote the Boolean XOR gate with $\oplus$ in the paper. Likewise we use $\leftrightarrow$ (or $=$, or $\equiv$) for bi-equivalence and $A \models B$ to mean that models of A are also models of B. A CNF ϕ can trivially be represented as a Boolean circuit σ as follows: for every $C \in \phi$, σ has $|vars(C)|$ number of OR-gates. Then, σ has $m-1$ AND-gates where m is the number of clauses $\in \phi$.

2.3 Proof Systems

A *proof system* [12] is a polynomial-time function that maps proofs to theorems, where the set of theorems is some fixed language L. A proof system is sound if its image is contained in L and complete if L is contained in its image. A proof system takes in strings as its inputs. Let π be such a proof we denote its *size*, i.e. the string length by $|\pi|$. Given two proof systems f and g for the same language L. We that the f *p-simulates* g, when there is a polynomial time function r that maps g-proofs to f-proofs such that $g(\pi_1) = f(r(\pi_1))$. f and g are said to be *p-equivalent* if they both p-simulate each other,f and g are incomaparable if neither of them p-simulates the other. We say that f is exponentially stronger than g, if f p-simulates g but g does not p-simulate f.

Conventionally we may take L to be the set of propositional tautologies (as in Section 2.3.1). For propositional model counting, we take L as the set of all pairs $(\phi, \#_{\text{models}}(\phi))$, where ϕ is any propositional formula. We refer to a #SAT proof of $(\phi, \#_{\text{models}}(\phi))$ as a proof of ϕ.

▶ **Definition 1** (Closure under restrictions [23]). *A proof system P is closed under restrictions if for every P-proof π of a CNF formula ϕ and any partial assignment α to $vars(\phi)$, there exists a P-proof π' of $\phi|_\alpha$ such that $|\pi'| \leq p(|\pi|)$ for some polynomial p. In addition, there exists a polynomial time procedure (w.r.t. $|\pi|$) to extract π' from π.*

A similar definition called "closure under conditioning" exists in the knowledge-compilation domain [14, Definition 3]. Precisely, a knowledge representation structure S (like DNNF, POG, etc) is closed under conditioning if from an S structure T and an assignment α to $vars(T)$, another S structure T' can be computed just by replacing all occurrences of free variables by α wherever defined. Additionally T' should be equivalent to $T \wedge \alpha$.

2.3.1 Propositional Proof Systems

Resolution [25] is arguably the most studied propositional proof system. It has the rule $\frac{(C \vee x) \quad (D \vee \overline{x})}{(C \cup D)}$ where C, D are clauses and x is a variable. Resolution refutation ρ of CNF ϕ is a derivation of $\perp$ using the above rule. Resolution is known to be closed under restrictions.

Frege systems [17] are important propositional proof systems. They consist of a sound and complete set of axioms and rules where any variable can be substituted by any formula. All Frege systems are p-equivalent [12]. Figure 2 gives one example of a Frege system.

$$\frac{}{1} \qquad \frac{}{x_1 \to (x_2 \to x_1)} \qquad \frac{}{((x_1 \to 0) \to 0) \to x_1}$$

$$\frac{}{(x_1 \to (x_2 \to x_3)) \to ((x_1 \to x_2) \to (x_1 \to x_3))} \qquad \frac{x_1 \qquad x_1 \to x_2}{x_2}$$

Figure 2 A Frege system for connectives $\to, 0, 1$.

Extended Frege (eFrege) [12] allows the introduction of new variables as well as all Frege rules. Simultaneously we can imagine it as a Frege system where lines are circuits instead of formulas, or as Substitution Frege, where derived tautologies can be generalised. eFrege is also p-equivalent to **DRAT** (Deletion Resolution Asymmetric Tautology) [21], a practical proof-format widely used in certifying SAT solvers. In Figure 3 we show that eFrege sits at the top of the simulation hierarchy of propositional proof systems. In fact eFrege can simulate any proof system as long as there is a short proof of the reflection principle of said proof system [20].

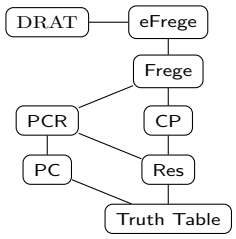

Figure 3 The p-simulation hierarchy of propositional proof systems [12].

When showing that eFrege has short proofs for complicated tautologies, it does not help us to be committed to one strictly defined proof system. Instead, we can use the fact that it can simulate many different proof systems such as Resolution, Cutting planes, Polynomial calculus and Truth Tables. Any tautology that has a short proof in any weaker system will also have a short proof in eFrege.

3 Circuit Linear Induction Proposition (CLIP) Proof Framework

In this section, we define a propositional model counting proof framework (CLIP+$\mathcal{P}$) for any propositional proof system $\mathcal{P}$. Given a CNF ϕ over n variables, a CLIP+$\mathcal{P}$ proof consists of a Boolean circuit ξ (denoted as a *cumulator*) which outputs the total number of models of ϕ from complete assignment $\mathbf{0}$ to a given complete assignment $num(\alpha)$ (denoted as $C_{models}(\phi, \alpha)$, Definition 2). Clearly, when $num(\alpha) = \mathbf{2}^n - \mathbf{1}$, $C_{models}(\phi, \alpha) = \#_{\text{models}}(\phi)$. In addition, the CLIP+$\mathcal{P}$-proof also requires a $\mathcal{P}$-proof of a statement which carefully encodes the correctness of cumulator ξ using the induction-principle (see Definition 5). We need the following definitions.

▶ **Definition 2** ($C_{models}(\phi, \alpha)$). *Let ϕ be an CNF formula, fix an order among vars(ϕ). For any complete assignment α to vars(ϕ), the cumulative number of models of CNF ϕ w.r.t. α (denoted by $C_{models}(\phi, \alpha)$) is the number of models of ϕ between assignment $\mathbf{0}$ to assignment num(α). In other words $C_{models}(\phi, \alpha) := \Sigma_{num(\beta) \leq num(\alpha)} \mathbb{1}_{\phi(\beta)}$, where $\mathbb{1}_{\phi(\beta)}$ is the indicator function for when β is a model of ϕ.*

▶ **Definition 3** (Cumulator). *A cumulator for a CNF ϕ over n variables is a Boolean multi-circuit $\xi(\alpha)$ which takes as input a complete assignment α to vars(ϕ) (as n binary bits) and calculates the cumulative number of models of ϕ, i.e. $C_{models}(\phi, \alpha)$ outputted as $n+1$ binary bits. As a result, when α is the last assignment (i.e. $num(\alpha) = \mathbf{2^n - 1}$), $\xi(\alpha)$ outputs the total number of models of ϕ, we denote this as the final output of ξ.*

A trivial cumulator for ϕ would be to keep a counter and given any α, input every assignment from $\mathbf{0}$ to $num(\alpha)$ into the trivial Boolean circuit representing ϕ. If an assignment is a model then increment the counter. This will take $\mathcal{O}(2^{|vars(\phi)|})$ computations in the case of α being the last assignment.

Consider a CNF ϕ and let k be its number of models. Given a cumulator $\xi(\alpha)$ for ϕ, the correctness of the cumulator can be encoded inductively as follows:

For the base case when $num(\alpha) = \mathbf{0}$, we need to verify that the following is satisfied: $(\phi(\alpha) \ \wedge \ ||\xi(\alpha) = \mathbf{1}||) \ \vee \ (\overline{\phi(\alpha)} \ \wedge \ ||\xi(\alpha) = \mathbf{0}||)$. This covers the case that if the first assignment is a model for ϕ then the cumulator should return $\mathbf{1}$, else a $\mathbf{0}$.

For the inductive step when $num(\alpha) = num(\beta) + \mathbf{1}$, the following should be satisfied $(\phi(\alpha) \ \wedge \ ||\xi(\alpha) = \xi(\beta) + \mathbf{1}||) \ \vee \ (\overline{\phi(\alpha)} \ \wedge \ ||\xi(\alpha) = \xi(\beta)||)$. This covers the case that if the next assignment α after β is a model of ϕ, then the cumulator should increment its output by 1. Otherwise, the cumulator should output the same number under both assignments.

For the final case when $num(\alpha) = \mathbf{2}^{|vars(\phi)|} - \mathbf{1}$, it should be true that $\xi(x) = k$. This covers the case that the cumulator computes the correct total number of models of ϕ.

It is clear to see that if all of the above cases are true, the cumulator ξ is proven to be a correct cumulator of ϕ. From the above discussion, one can encode the correctness of ξ as the following statement ($||\cdot||$ encloses the arithmetic comparisons needed):

$||num(\alpha) = \mathbf{0}|| \rightarrow \big((\phi(\alpha) \wedge ||\xi(\alpha) = \mathbf{1}||) \vee (\neg\phi(\alpha) \wedge ||\xi(\alpha) = \mathbf{0}||)\big)$
$\wedge ||num(\alpha) = num(\beta) + \mathbf{1}|| \rightarrow \big((\phi(\alpha) \wedge ||\xi(\alpha) = \xi(\beta) + \mathbf{1}||) \vee (\neg\phi(\alpha) \wedge ||\xi(\alpha) = \xi(\beta)||)\big)$
$\wedge ||num(\alpha) = \mathbf{2}^{|vars(\phi)|} - \mathbf{1}|| \rightarrow ||\xi(x) = k||.$

To convert this into a purely propositional statement, we need Boolean circuits to implement the arithmetic conditions $||x = y||$ and $||x = y + \mathbf{1}||$ for any integers x, y. We define polynomial sized Boolean circuits for the same as $E(x, y)$ and $T(x, y)$ respectively in Definition 4 below.

▶ **Definition 4.** *Let Z be a set of variables of size n, and let γ and δ be assignments to Z. For pairs of individual variables a, b, use $a = b$ to denote $(\neg a \vee b) \wedge (a \vee \neg b)$. We can encode polynomial size propositional circuits:*

- *$E(\gamma, \delta)$, that denotes $num(\gamma) = num(\delta)$: $E_0(\gamma, \delta) := (\gamma_0 \leftrightarrow \delta_0)$. For $1 \leq i < n$, $E_i(\gamma, \delta) := (\gamma_i \leftrightarrow \delta_i) \wedge E_{i-1}(\gamma, \delta)$. $E(\gamma, \delta) := E_{n-1}(\gamma, \delta)$.*
- *$T(\gamma, \delta)$, that denotes $num(\gamma) = num(\delta) + \mathbf{1}$ using an intermediate definition S that will denote the successor function. For $0 \leq i < n$ and accepting the empty conjunction as true, $S(\delta)_i := \neg(\delta_i \leftrightarrow \bigwedge_{j \geq 0}^{j<i} \delta_j)$. $T(\gamma, \delta) := E(\gamma, S(\delta)) \wedge \bigvee_{i \geq 0}^{i<n} \overline{\delta}_i$.*

▶ **Definition 5** (CLIP+$\mathcal{P}$). *For every propositional proof system $\mathcal{P}$, the CLIP+$\mathcal{P}$ system for #SAT is a cumulator ξ for a CNF ϕ along with its correctness presented as a valid $\mathcal{P}$-proof of the following linear induction proposition statement lip(ξ).*

Let A and B be two disjoint copies of the variables in ϕ. The following is a tautology in the variables of $A \cup B$:

$$\mathit{lip}(\xi) :=$$
$$E\big(A, num^{-1}(\mathbf{0})\big) \to \Big(\Big(\overline{\phi(A)} \to E\big(\xi(A), num^{-1}(\mathbf{0})\big)\Big) \wedge \Big(\phi(A) \to T\big(\xi(A), num^{-1}(\mathbf{0})\big)\Big)\Big) \wedge$$
$$T(B, A) \to \Big(\Big(\overline{\phi(B)} \to E\big(\xi(B), \xi(A)\big)\Big) \wedge \Big(\phi(B) \to T\big(\xi(B), \xi(A)\big)\Big)\Big) \wedge$$
$$E\big(A, num^{-1}(\mathbf{2}^{|vars(\phi)|} - \mathbf{1})\big) \to E\big(\xi(A), num^{-1}(k)\big).$$

The existence of a valid $\mathcal{P}$-proof of $\mathsf{lip}(\xi)$, ensures that ξ is correct and the final output k of ξ is the correct number of models of ϕ. Note that in a technical sense the proof of inductive step (i.e. line 2 in $\mathsf{lip}(\xi)$) is sufficient to verify the cumulator ξ, as the base and final case can be managed in the checker.

▶ **Theorem 6.** *If $\mathcal{P}$ is a propositional proof system then* CLIP$+\mathcal{P}$ *is a propositional model counting proof system.*

Proof. CLIP$+\mathcal{P}$ is sound and complete for #SAT as a trivial cumulator always exists for any ϕ and the propositional proof system $\mathcal{P}$ is sound and complete. Note that for a refutational proof system $\mathcal{P}'$, CLIP$+\mathcal{P}'$ can include the correctness of ξ by including a $\mathcal{P}'$-refutation of $\overline{\mathsf{lip}(\xi)}$ from the above definition. For polynomial time checkability, we perform three steps: a) Verify that ξ is indeed a circuit. b) Using ξ, generate $\mathsf{lip}(\xi)$ once again, to make sure it matches (where $\mathcal{P}$ does not accept circuits a canonical translation, i.e. a Tseitin transformation is needed). c) Verifying the $\mathcal{P}$ proof. ◀

▶ **Theorem 7.** CLIP$+$eFrege *has a super-polynomial lower bound only if* eFrege *has a super-polynomial lower bound or* $\mathsf{P}^{\#\mathsf{P}} \not\subseteq \mathsf{P/poly}$.

Proof. Suppose there is a family $(\phi_n)_{n\geq 0}$ of propositional formulas that are a super-polynomial lower bound to CLIP+eFrege. Let $f_{n,i}$ be the i^{th} bit of the cumulative function for ϕ_n. $(f_{n,i})_{n\geq 0}^{0\leq i\leq |vars(\phi_n)|}$ is a $\mathsf{P}^{\#\mathsf{P}}$ family. Finding the value of the cumulator at assignment α can be found by adding a constraint to ϕ that the only acceptable models are less than or equal to α and querying for the number of models.

Now suppose $\mathsf{P}^{\#\mathsf{P}} \subset \mathsf{P/poly}$, then there are polynomial size circuits for each $f_{n,i}$ and thus a polynomial size cumulator ξ_n for each ϕ_n. For each n, $\mathsf{lip}(\xi_n)$ is also polynomial size in ϕ_n. Thus the family $(\mathsf{lip}(\xi_n))_{n\geq 0}$ is super-polynomial lower bound for eFrege. ◀

4 CLIP+eFrege simulates existing #SAT proof systems

In this section, we give an important CLIP+eFrege p-simulation technique for any #SAT proof systems which are closed under restrictions and have short eFrege proofs of the properties of restriction (Definition 15). To be precise, we show that CLIP+eFrege can p-simulate any model counting proof system $\mathcal{P}$ which obey the following conditions:

- **I.** The polynomial-time ability to extract circuits θ from a $\mathcal{P}$-proof π of CNF ϕ over n variables $(x_{n-1} \dots x_0)$ that calculate closure under restrictions for any given (partial) assignment α. That is, $\theta : \alpha \to \#\mathrm{SAT}(\phi|_\alpha)$ (Definition 8).
- **II.** $\mathcal{P}$ has short eFrege proofs for properties of restriction (Definition 15) that confirm the correctness of closure under restrictions in $\mathcal{P}$.

Let us formally define the circuit θ used in above conditions.

▶ **Definition 8.** *Let ϕ be a CNF on n variables and α be a (partial) assignment of length $n-i$. We define $\theta^i(\alpha)$ to be a Boolean circuit that returns $\#_{models}(\phi|_\alpha)$. Also, $\theta^i_j(\alpha)$ is a circuit that returns the j^{th} bit of $\theta^i(\alpha)$.*

In the upcoming subsections, we give the complete simulation technique. Recall that CLIP+eFrege proof consists of a cumulator ξ and a eFrege-proof of validity of the propositional "lip" statement which encodes the correctness of ξ. Using the condition-I above, in Section 4.1 we derive the cumulator ξ for ϕ (Part 1 of our simulation technique). In Section 4.2, we use the condition-II to derive the eFrege-proof of lip(ξ) (Part 2 of our simulation technique).

4.1 Simulation Technique (Part 1) : Cumulator Extraction

In this section, we give a general framework of extracting efficiently a cumulator from the proofs of existing propositional model counting proof systems. We need the following definition.

▶ **Definition 9** (Disjoint binary partial assignment cover (Cov(J_1,J_2))). *Let J_1, J_2 be integers representing some complete assignments to variables $X :=< x_{n-1}, ..., x_0 >$ in this order. The cover Cov(J_1,J_2) is a set of partial assignments to X which are non-overlapping and together cover the entire assignment space between J_1 and J_2 inclusive of both.*

For instance, let $X := \{x_3, x_2, x_1, x_0\}$, $J_1 := \mathbf{5}$ and $J_2 := \mathbf{15}$. One possible Cov(J_1,J_2):= $\big\{\{x_3 = 1\}, \{x_3 = 0, x_2 = 1, x_1 = 1\}, \{x_3 = 0, x_2 = 1, x_1 = 0, x_0 = 1\}\big\}$. Observe that the first partial assignment (i.e. $\{x_3 = 1\}$) is covering all assignments from $[\boldsymbol{8}, \boldsymbol{15}]$. Similarly the second and third partial assignments are covering the assignments $[\boldsymbol{6}, \boldsymbol{7}]$ and $\mathbf{5}$ respectively. Another possible Cov(J_1,J_2):= $\big\{\{x_2 = 1, x_0 = 1\}, \{x_3 = 1, x_2 = 0\}, \{x_3 = 1, x_2 = 1, x_0 = 0\}, \{x_3 = 0, x_2 = 1, x_1 = 1, x_0 = 0\}\big\}$ which cover assignments $\{\boldsymbol{5}, \boldsymbol{7}, \boldsymbol{13}, \boldsymbol{15}\}, \{\boldsymbol{8}, \boldsymbol{9}, \boldsymbol{10}, \boldsymbol{11}\}, \{\boldsymbol{12}, \boldsymbol{14}\}$ and $\{\boldsymbol{6}\}$ respectively.

Let us now outline the general extraction technique.

Cumulator Extraction Technique. Let $\mathcal{P} \in \{\mathsf{MICE}, \mathsf{KCPS(\#SAT)}, \mathsf{CPOG}\}$ be a propositional model counting proof system. Consider a CNF ϕ over n variables and its $\mathcal{P}$-proof π. In order to efficiently extract a correct cumulator ξ for ϕ, we follow the following steps:

1. Show that $\mathcal{P}$ is closed under restrictions (see Definition 1). That is, show that $\mathcal{P}$ obeys condition-I from above.
2. For any complete assignment J to *vars*(ϕ), find the set of non-overlapping partial assignments (to *vars*(ϕ)) which cover the entire assignment space from assignment-$\mathbf{0}$ up to assignment-J (i.e Cov($\mathbf{0}$,J) see Definition 9).
   Using Fenwick's idea [15, 26], it is easy to compute Cov($\mathbf{0}$,J) for any complete assignment J (Lemma 10). Moreover, |Cov(0,J)|$\leq n$.
3. For each partial assignment $\alpha \in$ Cov($\mathbf{0}$,J), restrict π with α and consider the $\mathcal{P}$-proof π' of CNF $\phi|_\alpha$. Observe that π' and $\theta^i(\alpha)$ agree on the value of $\#SAT(\phi|_\alpha)$ where $i = n - |\alpha|$. Since $\mathcal{P}$ is closed under restrictions, this step takes $\mathcal{O}(|\pi|)$ time for every α.
4. Finally add the number of models returned by all the π' proofs obtained in the above step. (This step will need a full-adder circuit as integers are represented as $(n+1)$-bit numbers).

This process will return $\xi(J)$ which computes $C_{models}(\phi, J)$ and takes $\mathcal{O}(n.|\pi|)$ time.

We prove Step-1 of our simulation technique individually for existing proof system CPOG in Section 5 (Lemma 23). For Step-2, consider the following lemma.

▶ **Lemma 10.** *Given an input size n, and binary integer $\mathbf{0} \leq J < 2^n$. There is a polynomial time algorithm in n that returns a disjoint binary partial assignment cover for $[\mathbf{0}, J]$ (Cov($\mathbf{0}$,J)) with at most n many partial assignments.*

We have proved Lemma 10 by using a deterministic Fenwick-based algorithm in Appendix A.

Note that extracting the cumulator is not enough for the full CLIP simulation, because CLIP proofs also consist of the validity proof of the lip statement. However, with an access to an NP-oracle, the validity of the lip statement can be obtained in one step due to the correctness of our simulation technique. We call such a system as CLIP^{NP}. Thus the efficient cumulator extraction shows that CLIP^{NP} p-simulates any #SAT system which is closed under restrictions. Observe that CLIP^{NP} is a proof system only if P = NP.

In the upcoming sections, we give full CLIP framework simulations of all the existing #SAT proof systems using the powerful eFrege system for validating the lip statement.

Recall that for the cumulator extraction, we used the concepts of Fenwick assignments. In upcoming proofs, we also need some additional results on Fenwick assignments. We finish this subsection with these results before proceeding to the part 2 of our simulation technique.

4.1.1 Formalising Fenwick Assignments

In Lemma 10, we show how to compute the partial assignment cover of a given complete assignment α with $|\text{Cov}(\mathbf{0},\alpha)| \leq |\alpha|$. In this section, we formalise it as a Boolean circuit (Definition 11) which outputs a partial assignment cover for any given complete assignment α. We call the output Cov($\mathbf{0}$,J) of the Boolean circuit as Fenwick assignments. We further show that eFrege can handle a few essential properties of Fenwick assignments. We use these in the next section for part-2 of the simulation technique.

▶ **Definition 11** (Fenwick Assignments)**.** *Let α be a complete assignment to n variables with ordering $< x_{n-1} \dots x_0 >$. For every $i, 0 \leq i \leq n$, we define an existence function $e_i(\alpha)$ and the set of initial assignment bits $f_{i,j}$ for $0 \leq i \leq j < n$ as follows:*

$$e_i(\alpha) = \begin{cases} 1 & \alpha(x_i) \wedge \bigvee_{k\geq 0}^{k<i} \overline{\alpha(x_k)} \text{ and } 0 < i < n \\ 1 & \overline{\alpha(x_i)} \wedge \bigwedge_{k\geq 0}^{k<i} \alpha(x_k) \text{ and } 0 \leq i < n \\ 1 & \bigwedge_{k\geq 0}^{k<n} \alpha(x_k) \text{ and } i = n \\ 0 & \text{otherwise} \end{cases} \qquad f_{i,j}(\alpha) = \begin{cases} 1 & \alpha(x_j) \text{ and } j > i \\ 0 & \text{otherwise} \end{cases}$$

We denote for $0 \leq i < n$, $f_i(\alpha) = \{f_{i,j} | i \leq j < n\}$ as the i^{th} partial assignment for α (note that f_n is the empty assignment and needs no variables to be defined). For a complete assignment α, Fenwick assignments are $\{f_i(\alpha) | e_i(\alpha) = 1, 0 \leq i \leq n\}$. Here, $e_i(\alpha)$ can be seen as a single-bit value that indicates if there is an initial assignment defined on $n - i$ variables in the Fenwick assignments of α. Similarly, $f_{i,j}(\alpha)$ is the value of x_j in the i^{th} partial assignment corresponding to $e_i(\alpha)$ in Fenwick assignments of α.

Below we give an example of how we represent Fenwick assignments as in Definition 11.

▶ **Example 12.** Let $n = 4$ and $J = \mathbf{12}$. Let the variables be lexicographical ordered as $< x_3, \dots, x_0 >$. The corresponding Cov($\mathbf{0}$,$\mathbf{12}$) is the following set of partial assignments: $\Big\{\{x_3 = 1, x_2 = 1, x_1 = 0, x_0 = 0\},\ \{x_3 = 1, x_2 = 0\},\ \{x_3 = 0\}\Big\}$.

The Fenwick circuits have the following values: $e_0(\mathbf{12}) = 1,\ e_1(\mathbf{12}) = 0,\ e_2(\mathbf{12}) = 1,\ e_3(\mathbf{12}) = 1,\ e_4(\mathbf{12}) = 0$. This indicates that there are 3 partial Fenwick assignments in Cov($\mathbf{0}$,$\mathbf{12}$) ending at x_0, x_2 and x_3 respectively. The exact assignments are computed as follows:

$f_{0,3}(\mathbf{12}) = 1,\ f_{0,2}(\mathbf{12}) = 1,\ f_{0,1}(\mathbf{12}) = 0,\ f_{0,0}(\mathbf{12}) = 0 \quad \to \{x_3 = 1, x_2 = 1, x_1 = 0, x_0 = 0\}$
$f_{2,3}(\mathbf{12}) = 1,\ f_{2,2}(\mathbf{12}) = 0 \quad \to \{x_3 = 1, x_2 = 0\}$
$f_{3,3}(\mathbf{12}) = 0 \quad \to \{x_3 = 0\}$

We will be able to prove few properties of how Fenwick assignments change as $num(\alpha)$ increases slowly (see Appendix B). Step 3,4 of our simulation technique require restricting the θ circuit with all the Fenwick assignments of some complete assignment α and adding them up to get $\xi(\alpha)$. Using the formal definition of the Fenwick assignments from Definition 11, we have the following.

▶ **Definition 13.** *For a complete assignment α on n variables, we define the vector of Boolean variables $\xi(\alpha)$ as the following sum:*

$$(e_n(\alpha) \wedge \theta^n(f_n(\alpha))) + (e_{n-1}(\alpha) \wedge \theta^{n-1}(f_{n-1}(\alpha))) + \cdots + (e_0(\alpha) \wedge \theta^0(f_0(\alpha)))$$

This circuit ξ is the required cumulator.

4.2 Simulation Technique (Part 2): eFrege certification of the cumulator

Recall that, for a full CLIP+eFrege simulation, the proof system must have short eFrege proofs of the properties of restriction, i.e. condition-II from Section 4. Let us formally define the properties of restriction in Definition 15 which are based on the following simple observations of partial assignments.

▶ **Observation 14.** *Let ϕ be a CNF on variables $< x_{n-1} \dots x_0 >$ (used in that lexicographic ordering in CLIP). Let α be a partial assignment defined on $x_{n-1} \dots x_i$, undefined on $x_{i-1} \dots x_0$. Given a $\{0,1\}$-value b, let $\alpha_b := \alpha \cup \{x_{i-1} = b\}$. Then, $\#_{models}(\phi|_\alpha) = \#_{models}(\phi|_{\alpha_0}) + \#_{models}(\phi|_{\alpha_1})$. If α is a complete assignment, $\#_{models}(\phi|_\alpha) = \mathbb{1}_\phi(\alpha)$.*

▶ **Definition 15** (Properties of Restriction)**.** *Let S be a propositional model counting proof system and ϕ be a CNF on n variables ($< x_{n-1} \dots x_0 >$). Suppose ϕ has an S-proof π with θ being the associated circuit for restriction. We consider the following properties for all assignments α over $x_{n-1} \dots x_0$.*

1. *If α is a complete assignment: $\theta^0(\alpha) = \mathbb{1}_\phi(\alpha)$.*
2. *If α is a partial assignment defined on $x_{n-1} \dots x_i$: $\theta^i(\alpha) = \theta^{i-1}(\alpha_0) + \theta^{i-1}(\alpha_1)$*

▶ **Observation 16.** *Any propositional model counting proof system $\mathcal{P}$ which is closed under restrictions, satisfies the properties of restriction mentioned in Definition 15.*

Next we prove that short eFrege proofs of the properties of restriction can be used to give short eFrege-proofs of the lip statement from Definition 5.

▶ **Lemma 17.** *Suppose $\mathcal{P}$ is a propositional model counting proof system which is closed under restrictions. Let ϕ be a CNF and ξ be a cumulator obtained by using Fenwick assignments on the $\mathcal{P}$-proof of ϕ. If $\mathcal{P}$ has polynomial-sized* eFrege *proofs of the properties of restriction, then it has short* eFrege *proof of lip(ξ).*

Before presenting the detailed proof of Lemma 17, we briefly give the proof idea of it. For a CNF ϕ and any two consecutive assignments $\beta^2 = \beta^1 + 1$, we need to show that $\xi(\beta^2) = \xi(\beta^1) + \mathbb{1}_\phi(\beta^2)$. We show this in the following two cases: β^2 being odd or even. In the case of β^1=odd and β^2=even, we use the property of Fenwick assignments (Lemma 37, see Appendix B) that the Fenwick assignments of β^2 are the Fenwick assignment of β^1 and β^2. This directly implies $\xi(\beta^2) = \xi(\beta^1) + \mathbb{1}_\phi(\beta^2)$.

In the case of β^1 is even and β^2 is odd, we also relate the Fenwick assignments of the two total assignments. We use the property of Fenwick assignments (Lemma 38, see Appendix B) that the Fenwick assignment of β^2 is some common set of assignments β^* and a single partial assignment γ, γ is the common prefix of β^1 and β^2. We then show that using Observation 16 a linear number of times, we can prove that $\theta(\gamma)$ decomposes so that it implies that $\xi(\beta^2) = \xi(\beta^1) + \mathbb{1}_\phi(\beta^2)$.

Proof. Line-1 of the lip statement assumes the assignment $\alpha = \mathbf{0}$. From a simple computation, only $e_0(\mathbf{0}) = 1$ with the corresponding $f_0(\mathbf{0}) = \mathbf{0}$. From Definition 13, $\xi(\mathbf{0}) = \theta^0(\mathbf{0})$. From the properties of restriction $\theta^0(\mathbf{0}) = \mathbb{1}_\phi(\mathbf{0})$.

Line-3 of the lip statement assumes the assignment $\alpha = \mathbf{2^n - 1}$. Similarly, we can compute that only $e_n(\mathbf{2^n - 1}) = 1$ and $f_n(\mathbf{2^n - 1}) = \emptyset$. Then $\xi(\mathbf{2^n - 1}) = \theta^n(\emptyset)$. θ^n contains no restriction, so it is the intended answer for the entire model count.
The main part of CLIP is the inductive step for assignments $\beta^2 = \beta^1 + \mathbf{1}$.

For an even β^2, We can use the cases for short proofs in Lemma 38 (see Appendix B). We can argue that $e_i(\beta^1) \leftrightarrow e_i(\beta^2)$ and $e_i(\beta^1) \rightarrow (f_{i,j}(\beta^1) \leftrightarrow f_{i,j}(\beta^2))$ for $i > 0$. That is, all Fenwick assignments for β^1 and β^2 are the same except the one corresponding to $e_0(\beta^2)$. Additionally that $f_0(\beta^2) = \beta_2$. Along with associativity, this easily leads to $\xi(\beta^2) = \xi(\beta^1) + \mathbb{1}_\phi(\beta^2)$ in eFrege.

For an odd β^2, we can use the short proofs from Lemma 38 of the various cases (Appendix B). We take the maximum $p : 0 \leq p \leq n$ such that $\bigwedge_{j<p}^{j\geq 0} \beta_j^2$. We can derive (case-3a,3b) $e_j(\beta^1) \leftrightarrow e_j(\beta^2)$ and $f_{j,k}(\beta^1) \leftrightarrow f_{j,k}(\beta^2)$ for $j > p$. At $j = p$, we have that (case-2a,2b) $f_p(\beta^2) =$ the common prefix of β^1 and β^2 up to p (say γ). For $j < p$, we derive (case-1b) that γ is the prefix of all these Fenwick assignments. Further we show (case-1c,1d) that these assignments extend γ by $p - j - 1$ number of 1s and end with a 0. That is, $\{f_j(\beta^1)\}_{p>j\geq 0} = \gamma_0, \gamma_{10}, \gamma_{110}, \ldots, \beta^1$. Using the split property of restrictions for i times, we have $\theta(\gamma) = \sum_{j\leq p}^{j\geq 0}(e_j(\beta^1) \wedge \theta^j(f_j(\beta^1))) + \theta^0(\beta^2)$. Adding $f_{j,k}(\beta^1) \leftrightarrow f_{j,k}(\beta^2)$ for $j > p$ to the above gives us $\xi(\beta^2) = \xi(\beta^1) + \mathbb{1}_\phi(\beta^2)$. ◀

Next, we give a supplementary example of Lemma 17.

▶ **Example 18.** Let a CNF ϕ be defined on $n = 5$ variables and β^1, β^2 be $10010, 10011$ (i.e **18**, **19**) respectively. Let the variables be lexicographical ordered as $< x_4 \ldots x_0 >$. The corresponding Cov(**0**,**18**) and Cov(**0**,**19**) are: $\Big\{\{x_4 = 1, x_3 = 0, x_2 = 0, x_1 = 1, x_0 = 0\},\ \{x_4 = 1, x_3 = 0, x_2 = 0, x_1 = 0\},\ \{x_4 = 0\}\Big\}$ and $\Big\{\{x_4 = 1, x_3 = 0, x_2 = 0\},\ \{x_4 = 0\}\Big\}$ respectively.

The first zero of β^2 occurs for the second digit therefore we take $p = 2$. For $j > p$, clearly $e_4(\beta^1) = e_4(\beta^2)$ and $f_4(\beta^1) = f_4(\beta^2)$. There is one partial assignment that is in β^2 but not in β^1, it is $f_2(\beta^2)$ (i.e $\{x_4 = 1, x_3 = 0, x_2 = 0\}$). This is also the common prefix γ of both β^1 and β^2. Using the split property on γ twice, we have the following: $\theta^2(\gamma) = \theta^1(\gamma_0) + \theta^1(\gamma_1) = \theta^1(\gamma_0) + \theta^0(\gamma_{10}) + \theta^0(\gamma_{11})$. Adding $f_4(\beta^1) = f_4(\beta^2)$ to this implies that $\xi(\mathbf{19}) = \xi(\mathbf{18}) + \mathbb{1}_\phi(\mathbf{19})$.

For a model counting proof system $\mathcal{P}$ and a CNF ϕ with it's $\mathcal{P}$-proof π, we have a cumulator ξ from part 1 of our simulation technique. In Lemma 17, we also have an eFrege proof of lip(ξ). Therefore, we have the following.

▶ **Theorem 19.** *CLIP+eFrege p-simulates any model counting proof system which is closed under restrictions and has short eFrege proofs of the properties of restriction.*

In conclusion, for any propositional model counting proof system $\mathcal{P}$, part 2 of our simulation technique consists of the following step:

5. Show that $\mathcal{P}$ has short eFrege-proofs of the two properties of restriction (Definition 15). That is, show that $\mathcal{P}$ obeys condition-II from above.

5 CLIP framework simulation of CPOG

In this section, we apply our simulation technique to the CPOG (Certified Partitioned-Operation Graphs) [4] proof system. That is, we show that CPOG is closed under restrictions (Lemma 23) and admits easy eFrege proofs of the properties of restriction (Lemma 29). For a CNF formula ϕ, CPOG is a propositional weighted-model counting proof system which consists of a POG structure G (Definition 20) along with Resolution proofs of $\phi \leftrightarrow G$. It is well known that model counting is easy for POG [4, p. 16]. However, given a POG G and a CNF ϕ, verifying that $G \equiv \phi$ is hard. In this paper, we only study CPOG for the unweighted (standard) model counting. The missing proofs of this section are included in Appendix C.

The proofs in this section need to prove basic properties of arithmetic in short eFrege proofs which are considered to be academic folklore.

We first define the POG structure (Definition 20) and the CPOG proof system (Definition 22) which is based on POG from [4]. For an example of POG see [5].

▶ **Definition 20** (POG [4]). *A Partitioned-Operation Graph (POG) (say G) is a directed acyclic graph defined on n variables (say X). Each node v in a POG has an associated dependency set $\mathcal{D}(v) \subseteq X$ and a set of models $\mathcal{M}(v)$, consisting of all complete assignments that satisfy the formula represented by the POG rooted at v. The leaf nodes (with outdegree $= 0$) can be of the following:*

- *Boolean constants 0 or 1. Here, $\mathcal{D}(1) = \mathcal{D}(0) = 0$, $\mathcal{M}(0) = \emptyset$ and $\mathcal{M}(0) = \langle X \rangle$.*
- *Literal l for some variable x such that $vars(l) = x \in X$. Here, $\mathcal{D}(l) = x$, $\mathcal{M}(l) = \{\alpha \in \langle X \rangle \mid \alpha(x) \equiv l\}$.*

The rest of the nodes (internal nodes) can be of the following:

- *Decomposable AND-gate ($\wedge^p$) [1] with outgoing edges to $v_1, \ldots, v_k$ for $k > 1$. Here, $\mathcal{D}(\wedge^p) = \bigcup_{1 \le i \le k} \mathcal{D}(v_i)$ and $\mathcal{M}(\wedge^p) = \bigcap_{1 \le i \le k} \mathcal{M}(v_i)$. This node needs to follow the decomposable property namely, $\mathcal{D}(v_i) \cap \mathcal{D}(v_j) = \emptyset$ for every $i, j \in [k]$ and $i \neq j$.*
- *Deterministic OR-gate ($\vee^p$) with outgoing edges to v_1, v_2. Here, $\mathcal{D}(\vee^p) = \mathcal{D}(v_1) \cup \mathcal{D}(v_2)$ and $\mathcal{M}(\vee^p) = \mathcal{M}(v_1) \cup \mathcal{M}(v_2)$. This node needs to follow the deterministic property namely, $\mathcal{M}(v_1) \cap \mathcal{M}(v_2) = \emptyset$.*

The edges of G have an optional polarity to indicate if they need to be negated (polarity $= 1$) or not (polarity $= 0$). Here, $\mathcal{D}(\neg v) = \mathcal{D}(v)$ and $\mathcal{M}(\neg v) = \langle X \rangle - \mathcal{M}(v)$. Every POG has a designated root node r with indegree $= 0$ and models of the POG would be $\mathcal{M}(r)$.

The weighted model counting can be seen as a ring-evaluation problem for a commutative ring over rational numbers $\in [\mathbf{0}, \mathbf{1}]$. The ring evaluation problem takes a weight function $w(x) \in [\mathbf{0}, \mathbf{1}]$ for all variables $x \in X$ and computes the following:

$$R(v, w) = \Sigma_{\alpha \in \mathcal{M}(v)} \, \Pi_{l \in \alpha} \, w(l) \tag{1}$$

where $w(\overline{x}) = \mathbf{1} - w(x)$. For standard unweighted model-counting (i.e $|\mathcal{M}(v)|$), one can fix $w(x) = \frac{\mathbf{1}}{\mathbf{2}}$ for all $x \in X$ and $|\mathcal{M}(v)| = \mathbf{2}^{|\mathbf{X}|} \cdot R(v, w)$. The following properties of the ring evaluation function are well known.

[1] For simplicity, we use the same notations from [4]. Here, p stands for partitioned-operation formulas.

▶ **Proposition 21** ([4]). *Ring evaluations for operations $\neg$, $\wedge^p$ and $\vee^p$ satisfies the following for any weight function w: (i) $R(\neg v, w) = \mathbf{1} - R(v, w)$, (ii) $R(\bigwedge_{1\leq i\leq k}^{p} v_i, w) = \prod_{1\leq i\leq k} R(v_i, w)$, (iii) $R(v_1 \vee^p v_2, w) = R(v_1, w) + R(v_2, w)$.*

For a CNF ϕ, a CPOG proof π consists of a POG G such that $G \equiv \phi$. However, to make the proof easily verifiable, π explicitly has the proof that G is a POG and is equivalent to ϕ. We present the precise definition from [2] below.

▶ **Definition 22** (CPOG [2, 4]). *A CPOG proof π of ϕ is the tuple $(\mathcal{E}(G), \delta, \rho, \psi)$, where*

- *G is a POG such that $G \equiv \phi$ and $\mathcal{E}(G)$ is a clausal encoding of the POG G by defining an extension variable for every internal node of G.*
- *δ is the determinism proof for OR-gates which contains a Resolution proof of $\mathcal{E}(G) \wedge (v_1) \wedge (v_2)$ for every $\vee^p$-gate v with outgoing edges to v_1, v_2.*
- *ρ is the forward implication proof (i.e. $\phi \models G$) consisting of a Resolution proof of $\mathcal{E}(G) \wedge \phi \wedge (\overline{r})$.*
- *ψ is the reverse implication proof (i.e. $G \models \phi$) consisting of a Resolution proof of $\mathcal{E}(G) \wedge (r) \wedge \overline{C}$ for every clause $C \in \phi$.*

Observe that extension variables used in CPOG are restrictive as compared to those in eFrege.

▶ **Lemma 23.** CPOG *is closed under restrictions.*

This proof is fairly simple and we include it in Appendix C. This completes Step 1 of our simulation technique for CPOG. Thus we have the following:

▶ **Corollary 24.** *There is a polynomial time method of extracting a cumulator circuit from a* CPOG *proof.*

Now we are ready to prove that CPOG admits easy eFrege proofs for the properties of restriction. Recall, the weight function w is defined for all the variables (X) of POG G as $\frac{\mathbf{1}}{\mathbf{2}}$. In this case, the value of $R(r, w) = \mathbf{2}^{|\mathbf{X}|} \cdot |\mathcal{M}(r)|$. For an assignment α, conditioning the POG with α (Lemma 23) will give G' and the following will hold $R(r', w) = \mathbf{2}^{|\mathbf{X}-|\alpha||} \cdot |\mathcal{M}(r)|_\alpha|$.

Instead of changing the POG structure, we change the weight function as defined below to obtain the same model count as above i.e. $R(r, w_\alpha) = \mathbf{2}^{|\mathbf{X}-|\alpha||} \cdot |\mathcal{M}(r)|_\alpha|$.

▶ **Definition 25.** *Given a CNF ϕ and an initial assignment α defined on $x_{n-1} \dots x_i$ and undefined on $x_{i-1} \dots x_0$, we define the weight w_α which weighs variables according to the following*

$$w_\alpha(x_j) = \begin{cases} \mathbf{1} & j \geq i \ \&\ \alpha(x_j) = 1, \\ \mathbf{0} & j \geq i \ \&\ \alpha(x_j) = 0, \\ \frac{\mathbf{1}}{\mathbf{2}} & j < i. \end{cases}$$

Next, in Lemma 26, 27, 28, we prove some properties of $R(v, w_\alpha)$ for every node v of the POG recursively. We use the general properties of the ring function from Proposition 21 in these proofs. Informally, in Lemma 26, we show that if α was undefined on x and x is not in the dependency set of v (i.e. $x \notin \mathcal{D}(v)$), the value of R does not change when weight function is changed to w_{α_0} or w_{α_1} where $\alpha_b = \alpha \cup \{x = b\}$. In Lemma 27, we show that if x is in $\mathcal{D}(v)$, the values of R hold a weaker relation of $R(v, w_\alpha) = \frac{\mathbf{1}}{\mathbf{2}}(R(v, w_{\alpha_0}) + R(v, w_{\alpha_1}))$. We consider complete assignments α in Lemma 28 and prove that the function $R(v, w_\alpha)$ returns $\mathbf{1}$ if α is a satisfying assignment of the POG rooted at v and $\mathbf{0}$ otherwise. We use these Lemmas to prove that CPOG has easy eFrege proofs of the properties of restriction in Lemma 29. The detailed proofs of these lemmas are pushed to Appendix C.

▶ **Lemma 26.** *Let α be an initial partial assignment defined up to x_i where $i > 0$. We can prove in the structure of the POG that $R(v, w_\alpha) = R(v, w_{\alpha_0}) = R(v, w_{\alpha_1})$ when x_{i-1} is not in the dependency set of v. This proof can be formalised in a short* eFrege *proof.*

▶ **Lemma 27.** *Let α be an initial partial assignment defined up to x_i where $i > 0$. We can prove in the structure of the POG that $R(v, w_\alpha) = \frac{1}{2} \cdot (R(v, w_{\alpha_0}) + R(v, w_{\alpha_1}))$. Furthermore we can formalise this in short* eFrege *proofs.*

▶ **Lemma 28.** *For complete assignment α, we can prove using* eFrege *in the structure of the POG that $R(v, w_\alpha) = \mathbb{1}_v(\alpha)$.*

▶ **Lemma 29.** *CPOG has short eFrege proofs of $\theta(\alpha) = \mathbb{1}_\phi(\alpha)$, when α is a complete assignment, and $\theta(\alpha) = \theta(\alpha_0) + \theta(\alpha_1)$, when α is strictly initial and partial.*

Proof. We define $\theta^i(\alpha) = \mathbf{2^i} \cdot R(r, w_\alpha)$. Using Lemma 27 we can show for the root node r that $R(r, w_\alpha) = \frac{1}{2} \cdot (R(r, w_{\alpha_0}) + R(r, w_{\alpha_1}))$ when α is initial and strictly partial. This proves the second property of restriction.

Using Lemma 28 we can show that $R(r, \alpha) = \mathbb{1}_r(\alpha)$ when α is complete. Hence $\theta^0(\alpha) = \mathbb{1}_r(\alpha)$. Here for the first property of restriction, we still need to prove that $\mathbb{1}_r(\alpha) = \mathbb{1}_\phi(\alpha)$. For this, we use the Resolution proofs for $r \leftrightarrow \phi$ in the CPOG proof. Since eFrege p-simulates Resolution, these are easily converted to show $\mathbb{1}_r(\alpha) = \mathbb{1}_\phi(\alpha)$. ◀

This proves Step 5 of our simulation technique for CPOG. Therefore from our simulation technique part 1 and 2, we have the following.

▶ **Theorem 30.** CLIP+DRAT *p-simulates* CPOG.

In [2], the authors prove that CPOG is strictly stronger than the other existing proof systems (i.e. MICE, KCPS(#SAT)). Therefore we have the following.

▶ **Corollary 31.** CLIP+DRAT *p-simulates* MICE *and* KCPS(#SAT).

6 Exponential Improvement on Existing #SAT proof systems

In this section, we give easy CLIP+eFrege proofs for hard formulas of existing proof systems. Below, we give easy proofs of XOR-PAIRS in CLIP+eFrege system (Theorem 34). These formulas were previously proven to be hard for MICE [3, Theorem 23]. Later in Corollary 36, we give easy CLIP+eFrege proofs of some unsatisfiable formulas which are hard in MICE and KCPS(#SAT).

▶ **Definition 32** (XOR-PAIRS [3]). *Let $X = \{x_1, \ldots x_n\}$ and $Z = \{z_{1,1}, z_{1,2} \ldots, z_{n,n-1}, z_{n,n}\}$. $C^1_{ij} = (x_i \vee x_j \vee \bar{z}_{ij}), C^2_{ij} = (\bar{x}_i \vee x_j \vee z_{ij}), C^3_{ij} = (x_i \vee \bar{x}_j \vee z_{ij}), C^4_{ij} = (\bar{x}_i \vee \bar{x}_j \vee \bar{z}_{ij})$ $\phi(X, Z)$ contains $C^1_{ij}, C^2_{ij}, C^3_{ij}, C^4_{ij}$ for $i, j \in [n]$.*

The models of XOR-PAIRS are the assignments where $z_{i,j} = (x_i \oplus x_j)$ for all $i, j \in [n]$. Hence, $\#_{\text{models}}(\text{XOR-PAIRS}) = 2^n$. The family XOR-PAIRS is hard for proof systems MICE [3, Theorem 23]. We will show in Theorem 34 that these formulas are easy in CLIP+eFrege.

▶ **Definition 33.** *Fix an input length n, and let γ and δ be vectors of n variables. For pairs of individual variables a, b, use $a = b$ to denote $(\neg a \vee b) \wedge (\neg b \vee a)$. We can encode polynomial size propositional circuits: $L(\gamma, \delta)$, that denotes $num(\gamma) < num(\delta)$.*

▶ **Theorem 34.** CLIP+eFrege *has short proofs of XOR-PAIRS*

Proof. First we fix that all Z-bits are less significant than all X-bits, otherwise the cumulative function is affected by the variable ordering. We begin by arguing that the cumulative function for XOR-PAIRS is easy to compute. This comes from the fact the truth function itself behaves in a way that makes it amenable to counting, it only ever increases by one, once for each complete assignment to X. There is a function $p : 2^X \to 2^Z$ that maps the binary assignment α on X to the unique assignment in Z such that $\phi(\alpha, p(\alpha))$ for every α. We can construct a multi-output circuit P (a sequence of circuits $P_{i,j}$ for $i, j \in \{|X|\}$) for p, easily through $O(Z)$ many gates: $P_{i,j}(X) = (x_i \vee x_j) \wedge (\overline{x}_i \vee \overline{x}_j)$.

We then express the cumulative function in a cumulator circuit that we will use for CLIP.

$$\xi(\alpha, \beta) = \begin{cases} \alpha & \beta < P(\alpha) \\ \alpha + \mathbf{1} & \beta \geq P(\alpha) \end{cases}$$

Note that since $\xi(\alpha, \beta)$ outputs in binary we can actually express each digit as a Boolean circuit.

Now we have to argue why the remaining propositional proof is easy for eFrege. This is basically a number of tautological implications we have to show individually. The idea is to break each implication into a number of cases. Case analysis is typically easy for eFrege as it is just resolving with a disjunction of possibilities.

Base case. If $A_X = \mathbf{0}$, $P(A_X)$ always evaluates to $\mathbf{0}$. If A_Z is also $\mathbf{0}$, $\phi(A_X, A_Z)$ evaluates to true, while $L(A_Z, P(A_X))$ evaluates to false (because of strictness). This makes $\xi(\alpha, \beta)$ evaluate to the integer $\mathbf{1}$ (in other words $\xi(\alpha, \beta)_i = \mathbf{1}$ if and only if $i = n$). Each of these evaluations are shown in eFrege through the extension clauses. These will satisfy the two disjunctions that use the base case.

Inductive Step. Here we firstly argue that $\phi(B_X, B_Z) \leftrightarrow E(B_Z, P(B_X))$ has a short eFrege proof. We show that for each pair i, j the four clauses are implied by $(x_i \vee x_j) \wedge (\overline{x}_i \vee \overline{x}_j) \leftrightarrow z_{i,j}$. And then we show the four clauses show the truth table for$(x_i \vee x_j) \wedge (\overline{x}_i \vee \overline{x}_j) \leftrightarrow z_{i,j}$. The proof size is linear. If $B_X = A_X$ and $B_Z = A_Z + \mathbf{1}$, we make 3 cases.

1. Let $B_Z = P(B_X)$, we can get a short eFrege proof of $\neg L(B_Z, P(B_X))$ and $L(A_Z, P(A_X))$, and thus a proof of $T(\xi(B_X, B_Z), B_X)$ and $E(\xi(A_X, A_Z), A_X)$. We use $B_X = A_X$ to show $T(\xi(B_X, B_Z), \xi(A_X, A_Z))$. $\phi(B_X, B_Z) \leftrightarrow E(B_Z, P(B_X))$ is a proven tautology.
2. Let $B_Z < P(B_X)$, we get a short eFrege proof that $L(A_Z, P(A_X))$ is true, and thus a proof that $E(\xi(B_X, B_Z), B_X)$ and $E(\xi(A_X, A_Z), A_X)$. We use $B_X = A_X$ to show $E(\xi(B_X, B_Z), \xi(A_X, A_Z))$. $\phi(B_X, B_Z)$ falls into provable contradiction with $L(B_Z, P(B_X))$ by showing a bit must be different.
3. Let $B_Z > P(B_X)$, we can get a short eFrege proof of $\neg L(B_Z, P(B_X))$ and $\neg L(A_Z, P(A_X))$, and thus that $T(\xi(B_X, B_Z) = B_X + \mathbf{1})$ and $T(\xi(A_X, A_Z) = A_X + \mathbf{1})$. We use $B_X = A_X$ to show $E(\xi(B_X, B_Z), \xi(A_X, A_Z))$. $\phi(B_X, B_Z)$ contradicts $L(P(B_X), B_Z)$.

Now consider $||B_X = A_X + \mathbf{1}||$, $||B_Z = \mathbf{0}||$ and $||A_Z = \mathbf{2}^{|\mathbf{Z}|} - \mathbf{1}||$. Part of the trichotomy is impossible. We can prove $L(P(B_X), B_Z)$ fails when $||B_Z = \mathbf{0}||$. For the remaining cases we firstly prove that $\neg||\mathbf{2}^{|\mathbf{Z}|} - \mathbf{1} < P(A_X)||$ which is proven from the fact that one digit must be 0 to be less than. Therefore $\xi(A_X, A_Z) = A_X + \mathbf{1}$ in both cases.

1. Let $B_Z = P(B_X)$ then we can get a short eFrege proof that $L(B_Z, P(B_X))$ is false and so $\xi(B_X, B_Z) = B_X + \mathbf{1} = A_X + \mathbf{1} + \mathbf{1} = \xi(A_X, A_Z) + \mathbf{1}$. We can find an equality proof here. $\phi(B_X, B_Z) \leftrightarrow E(B_Z, P(B_X))$ is a tautology.

2. Let $L(B_Z, P(B_X))$ be true so $\xi(B_X, B_Z) = B_X = A_X + \mathbf{1} = \xi(A_X, A_Z)$. $\phi(B_X, B_Z)$ falls into provable contradiction with $L(B_Z, P(B_X))$.

For the final case, we use $\neg||\mathbf{2}^{|\mathbf{Z}|} - \mathbf{1} < P(A_X)||$, hence $\xi(\mathbf{2}^{|\mathbf{X}|} - \mathbf{1}, \mathbf{2}^{|\mathbf{Z}|} - \mathbf{1}) = \mathbf{2}^{|\mathbf{X}|}$. ◀

Let us briefly discuss about unsatisfiable formulas. That is, CNF formulas for which the model counts are $\mathbf{0}$. In [2], it has been shown that for unsatisfiable formulas, KCPS(#SAT) is p-equivalent to regular Resolution [2, Proposition 5.1] and MICE is p-equivalent to Resolution [2, Proposition 5.3]. In this paper, we observed the following for the unsatisfiable CNF formulas:

▶ **Proposition 35.** *For unsatisfiable formulas ϕ, if ϕ has an* eFrege *proof π of unsatisfiability, then ϕ has a* CLIP+eFrege *proof of linear size w.r.t. $|\pi|$.*

Proof. For a unsatisfiable CNF ϕ, assume that it has an easy eFrege-proof of unsatisfiability. We can have an easy CLIP+eFrege proof of ϕ as follows: The cumulator ξ for ϕ is a trivial circuit that only outputs "$\mathbf{0}$" for any input. For any two consecutive assignments i.e $\beta_2 = \beta_1 + \mathbf{1}$, the inductive statement of lip encodes that $\xi(\beta_2) = \xi(\beta_1) + \mathbb{1}_\phi(\beta_2)$. Therefore, the eFrege proof of lip statement needs only the unsatisfiability proof of ϕ (i.e. $\mathbb{1}_\phi(\beta_2) = \mathbf{0}$). ◀

This gives more separation results for unsatisfiable formulas which are hard for Resolution but easy for eFrege. That is, we have the following:

▶ **Corollary 36.** *The unsatisfiable formulas, PHP, clique-color and Random Parity have polynomial-size proofs [11, 6, 10] in* CLIP+eFrege *but require exponential-size proofs in [19, 22, 9]* MICE *and* KCPS(#SAT).

Note that the Clique-coloring principle [22, Definition 7.1] is well studied in proof complexity. Informally, it encodes that if a graph G has a clique of size k, then G needs at least k colors. PHP is the famous Pigeon hole principle which encodes that if there are n pigeons and $n-1$ holes, at least one hole has more than one pigeon in it. Random Parity formulas are contradictions expressing both the parity and non-parity on a set of variables.

7 Conclusion

We have introduced the CLIP framework for propositional model counting. We have demonstrated the advantages CLIP has by having an unrestricted underlying circuit format. Our approach here has been theoretical and no version of CLIP has been implemented.

The main checking task in CLIP proofs can use existing tools in SAT such as DRAT-trim [28]. We have given a p-simulation of all other #SAT proof systems, in theory this can be used to extract CLIP+eFrege (or CLIP+DRAT) proofs from #SAT solvers. However the number of arithmetic lemmas may make the complete programming of the extractor a difficult task. It could be compensated with assistance from a certifying SAT solver.

Future work should take into account weighted and projected model counting.

References

1 Olaf Beyersdorff, Ilario Bonacina, Leroy Chew, and Ján Pich. Frege systems for quantified boolean logic. *J. ACM*, 67(2):9:1–9:36, 2020. doi:10.1145/3381881.

2 Olaf Beyersdorff, Johannes Klaus Fichte, Markus Hecher, Tim Hoffmann, and Kaspar Kasche. The relative strength of #sat proof systems. In *27th International Conference on Theory and Applications of Satisfiability Testing, SAT 2024, August 21-24, 2024, Pune, India*, volume 305 of *LIPIcs*, pages 5:1–5:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICS.SAT.2024.5.

3 Olaf Beyersdorff, Tim Hoffmann, and Luc Nicolas Spachmann. Proof complexity of propositional model counting. In Meena Mahajan and Friedrich Slivovsky, editors, *26th International Conference on Theory and Applications of Satisfiability Testing, SAT 2023, July 4-8, 2023, Alghero, Italy*, volume 271 of *LIPIcs*, pages 2:1–2:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICS.SAT.2023.2.

4 Randal E Bryant, Wojciech Nawrocki, Jeremy Avigad, and Marijn J. H. Heule. Certified knowledge compilation with application to verified model counting. In *26th International Conference on Theory and Applications of Satisfiability Testing (SAT 2023)*, 2023. doi:10.4230/LIPICS.SAT.2023.6.

5 Randal E. Bryant, Wojciech Nawrocki, Jeremy Avigad, and Marijn J. H. Heule. Supplement to certified knowledge compilation with application to verified model counting, 2023. URL: https://zenodo.org/records/7966174.

6 Sam Buss and Neil Thapen. DRAT and propagation redundancy proofs without new variables. *Log. Methods Comput. Sci.*, 17(2), 2021. URL: https://lmcs.episciences.org/7400.

7 Samuel R. Buss. Towards NP-P via proof complexity and search. *Ann. Pure Appl. Log.*, 163(7):906–917, 2012. doi:10.1016/J.APAL.2011.09.009.

8 Florent Capelli. Knowledge compilation languages as proof systems. In *Theory and Applications of Satisfiability Testing - SAT 2019 - 22nd International Conference, SAT 2019, Lisbon, Portugal, July 9-12, 2019, Proceedings*, volume 11628 of *Lecture Notes in Computer Science*, pages 90–99. Springer, 2019. doi:10.1007/978-3-030-24258-9_6.

9 Leroy Chew, Alexis de Colnet, Friedrich Slivovsky, and Stefan Szeider. Hardness of random reordered encodings of parity for resolution and CDCL. In *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014*, pages 7978–7986. AAAI Press, 2024. doi:10.1609/AAAI.V38I8.28635.

10 Leroy Chew and Marijn J. H. Heule. Sorting parity encodings by reusing variables. In *Theory and Applications of Satisfiability Testing - SAT 2020 - 23rd International Conference, Alghero, Italy, July 3-10, 2020, Proceedings*, volume 12178 of *Lecture Notes in Computer Science*, pages 1–10. Springer, 2020. doi:10.1007/978-3-030-51825-7_1.

11 Stephen A Cook. A short proof of the pigeon hole principle using extended resolution. *Acm Sigact News*, 8(4):28–32, 1976. doi:10.1145/1008335.1008338.

12 Stephen A Cook and Robert A Reckhow. The relative efficiency of propositional proof systems. In *Logic, Automata, and Computational Complexity: The Works of Stephen A. Cook*, pages 173–192. ACM, 2023. doi:10.1145/3588287.3588299.

13 William J. Cook, Collette R. Coullard, and György Turán. On the complexity of cutting-plane proofs. *Discret. Appl. Math.*, 18(1):25–38, 1987. doi:10.1016/0166-218X(87)90039-4.

14 Adnan Darwiche. Decomposable negation normal form. *J. ACM*, 48(4):608–647, 2001. doi:10.1145/502090.502091.

15 Peter M. Fenwick. A new data structure for cumulative frequency tables. *Softw. Pract. Exp.*, 24(3):327–336, 1994. doi:10.1002/SPE.4380240306.

16 Johannes K Fichte, Markus Hecher, and Valentin Roland. Proofs for propositional model counting. In *25th International Conference on Theory and Applications of Satisfiability Testing (SAT 2022)*, 2022. doi:10.4230/LIPICS.SAT.2022.30.

17 Gottlob Frege. *Begriffsschrift, eine der arithmetischen nachgebildete Formelsprache der reinen Denkens, Halle.* Lubrecht & Cramer, 1879. English translation in: from Frege to Gödel, a source book in mathematical logic (J. van Heijenoord editor), Harvard University Press, Cambridge 1967.

18 Joshua A. Grochow and Toniann Pitassi. Circuit complexity, proof complexity, and polynomial identity testing: The ideal proof system. *J. ACM*, 65(6), November 2018. doi:10.1145/3230742.

19 Armin Haken. The intractability of resolution. *Theor. Comput. Sci.*, 39:297–308, 1985. doi:10.1016/0304-3975(85)90144-6.

20 Jan Krajíček. *Bounded Arithmetic, Propositional Logic, and Complexity Theory*. Cambridge University Press, New York, NY, USA, 1995.

21 Benjamin Kiesl, Adrián Rebola-Pardo, and Marijn JH Heule. Extended resolution simulates DRAT. In *Automated Reasoning: 9th International Joint Conference, IJCAR 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 14-17, 2018, Proceedings*, pages 516–531. Springer, 2018. doi:10.1007/978-3-319-94205-6_34.

22 Jan Krajícek. Interpolation theorems, lower bounds for proof systems, and independence results for bounded arithmetic. *J. Symb. Log.*, 62(2):457–486, 1997. doi:10.2307/2275541.

23 Meena Mahajan and Gaurav Sood. QBF merge resolution is powerful but unnatural. In *25th International Conference on Theory and Applications of Satisfiability Testing, SAT 2022, August 2-5, 2022, Haifa, Israel*, volume 236 of *LIPIcs*, pages 22:1–22:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPICS.SAT.2022.22.

24 Christos H. Papadimitriou. *Computational complexity*. Addison-Wesley, 1994.

25 John Alan Robinson. Theorem-proving on the computer. *Journal of the ACM*, 10(2):163–174, 1963. doi:10.1145/321160.321166.

26 Boris Ryabko. A fast on-line adaptive code. *IEEE Trans. Inf. Theory*, 38(4):1400–1404, 1992. doi:10.1109/18.144725.

27 Seinosuke Toda. PP is as hard as the polynomial-time hierarchy. *SIAM J. Comput.*, 20(5):865–877, 1991. doi:10.1137/0220053.

28 Nathan Wetzler, Marijn Heule, and Warren A. Hunt Jr. DRAT-trim: Efficient checking and trimming using expressive clausal proofs. In *SAT 2014*, volume 8561 of *Lecture Notes in Computer Science*, pages 422–429. Springer, 2014. doi:10.1007/978-3-319-09284-3_31.

A Missing proof and algorithm from Section 4.1

▶ **Lemma 10.** *Given an input size n, and binary integer $\mathbf{0} \leq J < 2^n$. There is a polynomial time algorithm in n that returns a disjoint binary partial assignment cover for $[\mathbf{0}, J]$ (Cov$(\mathbf{0},J)$) with at most n many partial assignments.*

Proof. Refer to Algorithm 1 for the exact procedure. The correctness of Algorithm 1 stems from the correctness of Fenwick trees [15] which efficiently computes the cumulative sum of numbers stored between any two indices of the array by visiting atmost logarithmic indices in the tree. Algorithm 1 similarly computes Cov$(\mathbf{0},J)$ by finding the least number of partial assignments to include such that they cover the entire range from $[\mathbf{0}, J]$ i.e. $log(2^n) = n$. ◀

B Missing lemmas and proofs from Section 4.1.1

▶ **Lemma 37.** *Assume $T(Y, X)$ (i.e. $Y = X + 1$) and $\neg Y_0$ then*

1. $e_0(Y) \wedge \neg e_0(X)$
2. *For $i > 0$, $e_i(X) \leftrightarrow e_i(Y)$*
3. *For $0 < i \leq j < n$, $e_i(X) \rightarrow (f_{i,j}(X) \leftrightarrow f_{i,j}(Y))$*
4. *For $0 \leq j < n$, $f_{0,j}(Y) \leftrightarrow Y_j$*

And we can prove these formally in eFrege *in short proofs even in the case that Y and X are vectors of variables (or extension variables).*

Proof. $\bar{Y_0} \wedge \bigwedge_{k\geq 0}^{k<0} Y_k$ is true hence $e_0(Y)$ can be shown via definition. Since, $Y_0 = X_0 \oplus \bigwedge_{k\geq 0}^{k<0} X_k$ and $\bigwedge_{k\geq 0}^{k<0} X_k$ is just 1 therefore X_0 is true and so $e_0(X)$ must be false and we can show this through a short derivation.

Algorithm 1 Fenwick tree [15] based algorithm to find Cov(0,J).

Require: $J < 2^n$
  function FENWICK-ASSIGNMENTS(int J, int n)
    int $\alpha := \{\}$, dash:= $\{\}$ /*α, dash are each a set of integers*/
    int $indx \leftarrow J + 1$ /*assignments $\in [0, 2^n - 1]$ but the Fenwick tree handles $[1, 2^n]$*/
    while $indx > 0$ **do**
      parent = $indx - (indx \;\&\; -indx)$ /*& is the bit-wise AND operator*/
      α.append(parent)
      dash.append($log(indx - \text{parent})$) /*records no. of variables to forget from α*/
      $indx \leftarrow$ parent
    end while
  return $\alpha' =$ process(α, dash)
  /*"process" function does the following: for $i \in |\alpha|$, $\alpha'[i]$ is the partial assignment obtained from $\alpha[i]$ after discarding "dash[i]" number of variables from the right end of the fixed ordering of variables*/
  end function

Take p to be the maximum such that $\bigwedge_{k\geq 0}^{k<p} X_i$ is true, we can prove such a maximum exists by exhibiting a disjunction. Then for $0 < i < p$, $e_i(X) = 0$. $Y_k = 0$ for $k \leq i$ by definition of T and so $\neg e_i(Y)$ by definition of $e_i(Y)$. For $i = p$, $X_i = 0$ while $\bigwedge_{k\geq 0}^{k<i} X_i$ and $Y_i = 1$ while $\bigvee_{k\geq 0}^{k<i} \bar{Y}_i$ so both $e_i(X)$ and $e_i(Y)$ are true. $f_{p,p}(X) = f_{p,p}(Y) = 0$ because i is not strictly greater than itself. For $j > p$, we have to show that X_j and Y_j are equal. Recall that $Y_j = X_j \oplus \bigwedge_{k\geq 0}^{k<j} X_k$, but since X_p is false, $\bigwedge_{k\geq 0}^{k<j} X_k = 0$ and so $Y_j = X_j$. Hence $f_{p,j}(X) = f_{p,j}(Y)$.

For $i > p$, if $e_i(X)$ is true then $X_i \wedge \bigvee_{k\geq 0}^{k<i} \bar{X}_k$ must be true. $\bigvee_{k\geq 0}^{k<i} \bar{Y}_k$ must also be true because Y_0 is true. Since $X_i = Y_i$ then $Y_i \wedge \bigvee_{k\geq 0}^{k<i} \bar{Y}_k$ is also true and so $e_i(Y)$ can be proven that way. Since $X_j = Y_j$ for $j \geq i$ then by definition $f_{i,j}(X) = f_{i,j}(Y)$.

By definition, for $j > 0$, $f_{0,j}(Y) = Y_j$ and $f_{0,0}(Y) = 0 = Y_0$. ◀

▶ **Lemma 38.** *Assume $T(Y, X)$ (i.e. $Y = X + 1$) and Y_0 then there is some maximum $p : 0 < p \leq n$ such that $\bigwedge_{j\geq 0}^{j<p} Y_j$. Further the following properties are true and have short formal eFrege proofs.*

1. a. $e_i(X) \wedge \neg e_i(Y)$ *for* $0 \leq i < p$
   b. $f_{i,j}(X) \leftrightarrow Y_j$ *for* $0 \leq i < p \leq j < n$
   c. $f_{i,j}(X)$ *for* $0 \leq i < j < p \leq n$
   d. $\neg f_{i,p}(X)$ *for* $0 \leq i < p$
2. a. $\neg e_p(X) \wedge e_p(Y)$ *if* $p < n$
   b. $f_{p,j}(X) \leftrightarrow f_{p,j}(Y)$ *for* $p \leq j < n$.
   c. $\neg f_{p,p}(X) \wedge \neg f_{p,p}(Y)$ *if* $p < n$
3. a. $e_i(X) \leftrightarrow e_i(Y)$ *for* $p < i < n$
   b. $f_{i,j}(X) \leftrightarrow f_{i,j}(Y)$ *for* $p < i \leq j < n$

Proof. Because $Y_0 = 1$, then by definition of T we can prove $X_0 = 0$, hence $\bigvee_{k\geq 0}^{k<i} \bar{X}_k$ is always true for $i > 0$. This means that through the definition of T, $Y_i = X_i$ for $i > 0$. This will be important for many items when $i > 0$.

For $i = 0$ we get $e_0(X)$ and $\neg e_0(Y)$ through definition and use of $Y_0 \wedge \neg X_0$. And for $0 < i < p$, $X_i = 1$ and since $\bigvee_{k\geq 0}^{k<i} \bar{X}_k$ is true $e_i(X)$ is true. However $\bigwedge_{j\geq 0}^{j\leq i} Y_j$ is true so $e_i(Y)$ is false. Through $Y_i = X_i$ we also get that for $0 \leq i < j$, $f_{i,j}(X) = X_j = Y_j$. For $j < p$ we specifically get $Y_j = 1$ and for $j = p$ we get $Y_j = 0$. This completes all cases from 1.

In case 2, $e_p(X)$ is false because $X_p = Y_p = 0$ and $\bigwedge_{k\geq 0}^{k<p} X_k$ is false. Likewise, $e_p(X)$ is true because $Y_p = 0$ and $\bigwedge_{k\geq 0}^{k<p} Y_k$ is true. $f_{p,p}(X) = f_{p,p}(Y) = 0$ by definition and $f_{p,j}(X) = f_{p,j}(Y)$, for $j > p$ because then $X_j = Y_j$.

In case 3, again $f_{i,j}(X) = f_{i,j}(Y)$, for $j \geq i > p$ because $X_j = Y_j$. We can also use $X_j = Y_j$ to show $e_i(X) = e_i(Y)$ because $\bigvee_{k\geq 0}^{k<i} \bar{X}_k$ and $\bigvee_{k\geq 0}^{k<i} \bar{Y}_k$ are now both true.

All these cases can be formalised in eFrege proofs due to their simplicity for each choice of p. One final important step is to create and prove disjunction over all possible p. ◀

C Missing proofs from Section 5

▶ **Lemma 23.** CPOG *is closed under restrictions.*

Proof. For a given CNF ϕ, a CPOG proof consists of a POG G and a Resolution proof of $\phi \leftrightarrow G$. A POG is closed under conditioning as for any partial assignment α: replace the inputs labelled by x with $\alpha(x)$ for every x assigned by α and the resulting structure is still a POG G'. This is because, constants are allowed in POG and the $\wedge^p$-nodes will remain decomposable since we are only reducing variables. Also, the $\vee^p$-gates will remain deterministic because if A and B has disjoint models, so are $A|_\alpha$ and $B|_\alpha$. The Resolution proof witness is closed under restrictions. The CPOG proof of $\phi \leftrightarrow G$ uses Resolution proofs which are closed under restrictions, it implies $\phi|_\alpha \leftrightarrow G|_\alpha$. ◀

▶ **Corollary 24.** *There is a polynomial time method of extracting a cumulator circuit from a* CPOG *proof.*

Proof. With a CPOG proof, given an assignment α, in polynomial time we can calculate the Cov($\mathbf{0}$,α) via Fenwick's method (Lemma 10) and use closure under restrictions to find the values for the sum. By formalising these steps into a circuit as in Definition 13 we get the cumulator circuit. ◀

▶ **Lemma 26.** *Let α be an initial partial assignment defined up to x_i where $i > 0$. We can prove in the structure of the POG that $R(v, w_\alpha) = R(v, w_{\alpha_0}) = R(v, w_{\alpha_1})$ when x_{i-1} is not in the dependency set of v. This proof can be formalised in a short* eFrege *proof.*

Proof. In all cases, except at leaves, the dependency set is the union of the dependency sets of its children.

Boolean leaf: $R(1, w_\alpha) = \mathbf{1}$ and $R(0, w_\alpha) = \mathbf{0}$ independent of α. Therefore the Lemma statement is easily derived in this case.

Variable leaf: Let the variable leaf be x_j. $R(x_j, w)$ takes the value of $w(x_j)$. If $x_{i-1} \notin \mathcal{D}(x_j)$, then either $j \geq i$ or $j < i - 1$. This is formalised in a tautological disjunction in eFrege.

In either case we show equality is easily derived. If $j \geq i$ then $w_\alpha(x_j) = \mathbf{1}$ when $\alpha(j) = 1$ which also extends to $\alpha_0(j) = 1$ and $\alpha_1(j) = 1$ in which case $w_{\alpha_0}(x_j) = \mathbf{1}$ and $w_{\alpha_1}(x_j) = \mathbf{1}$. Similarly all $w_\alpha(x_j) = w_{\alpha_0}(x_j) = w_{\alpha_1}(x_j) = \mathbf{0}$ when $\alpha(j) = 0$. If $j < i - 1$ then $\alpha, \alpha_0, \alpha_1$ are all undefined on x_j, so the weights are all $\frac{1}{2}$.

Negation: Using the induction hypothesis on the child node c, $R(c, w_\alpha) = R(c, w_{\alpha_0}) = R(c, w_{\alpha_1})$ and so $R(\neg c, w_\alpha) = \mathbf{1} - R(c, w_\alpha) = \mathbf{1} - R(c, w_{\alpha_0}) = \mathbf{1} - R(c, w_{\alpha_1})$ therefore $R(\neg c, w_\alpha) = R(\neg c, w_{\alpha_0}) = R(\neg c, w_{\alpha_1})$.

Partition Conjunction: Let C be the set of child nodes for $\wedge^p$. Using the induction hypothesis $R(c, w_\alpha) = R(c, w_{\alpha_0}) = R(c, w_{\alpha_1})$ for each child $c \in C$. We get the following equality for the products $\Pi_{c\in C} R(c, w_\alpha) = \Pi_{c\in C} R(c, w_{\alpha_0}) = \Pi_{c\in C} R(c, w_{\alpha_1})$. Thus $R(\bigwedge^p_{c\in C}, w_\alpha) = R(\bigwedge^p_{c\in C}, w_{\alpha_0}) = R(\bigwedge^p_{c\in C}, w_{\alpha_1})$.

Partition Disjunction: Let the child nodes of $\vee^p$ be c, d. Using the induction hypothesis $R(c, w_\alpha) = R(c, w_{\alpha_0}) = R(c, w_{\alpha_1})$ and $R(d, w_\alpha) = R(d, w_{\alpha_0}) = R(d, w_{\alpha_1})$.
$R(c \vee^p d, w_\alpha) = R(c, w_\alpha) + R(d, w_\alpha) = R(c, w_{\alpha_0}) + R(d, w_{\alpha_0}) = R(c \vee^p d, w_{\alpha_0})$. Similarly, it can be derived that $R(c \vee^p d, w_\alpha) = R(c \vee^p d, w_{\alpha_1})$.

Each inductive step involves a bounded application of implications using the definitions hence we get short eFrege proofs. ◀

▶ **Lemma 27.** *Let α be an initial partial assignment defined up to x_i where $i > 0$. We can prove in the structure of the POG that $R(v, w_\alpha) = \frac{\mathbf{1}}{\mathbf{2}} \cdot (R(v, w_{\alpha_0}) + R(v, w_{\alpha_1}))$. Furthermore we can formalise this in short* eFrege *proofs.*

Proof. If $x_{i-1} \notin \mathcal{D}(v)$, this directly holds from Lemma 26, along with arithmetic properties i.e. $a = \frac{\mathbf{1}}{\mathbf{2}} \cdot (a + a)$. So here we only consider the case that $x_{i-1} \in \mathcal{D}(v)$.

Variable leaf: Let the variable leaf be x_j. $x_{i-1} \in \mathcal{D}(v)$ implies that $j = i - 1$, then $R(v, w_\alpha) = \frac{\mathbf{1}}{\mathbf{2}}$. $R(v, w_{\alpha_0}) = \mathbf{1} \to R(v, w_{\alpha_1}) = \mathbf{0}$ and $R(v, w_{\alpha_0}) = \mathbf{0} \to R(v, w_{\alpha_1}) = 1$, in both cases they sum to 1 which is the right identity when multiplied with $\frac{1}{2}$.

Negation: Using the induction hypothesis on the child node c, i.e. $R(c, w_\alpha) = \frac{1}{2} \cdot (R(c, w_{\alpha_0}) + R(c, w_{\alpha_1}))$, it implies the following.

$$\begin{aligned} R(\neg c, w_\alpha) &= \mathbf{1} - R(c, w_\alpha) = \mathbf{1} - \tfrac{\mathbf{1}}{\mathbf{2}} \cdot (R(c, w_{\alpha_0}) + R(c, w_{\alpha_1})) \\ &= \mathbf{1} - \tfrac{\mathbf{1}}{\mathbf{2}} \cdot (\mathbf{1} - R(\neg c, w_{\alpha_0}) + \mathbf{1} - R(\neg c, w_{\alpha_1})) \\ &= \mathbf{1} - \tfrac{\mathbf{1}}{\mathbf{2}} \cdot (\mathbf{2} - R(\neg c, w_{\alpha_0}) - R(\neg c, w_{\alpha_1})) \\ &= \tfrac{\mathbf{1}}{\mathbf{2}} \cdot (R(\neg c, w_{\alpha_0}) + R(\neg c, w_{\alpha_1})). \end{aligned}$$

Partition Conjunction: Let C be the set of child nodes for $\wedge^p$. Observe that, $x_{i-1} \in \mathcal{D}(c^*)$ for exactly one child c^*. Along with multiplicative commutativity and associativity, we know the following from Proposition 21:

$R(\bigwedge^p_{c\in C} c, w_\alpha) = R(c^*, w_\alpha) \cdot \prod_{c\in\{C\setminus c^*\}} R(c, w_\alpha)$.

Using the induction hypothesis on c^* we get

$$= \tfrac{\mathbf{1}}{\mathbf{2}} \cdot (R(c^*, w_{\alpha_0}) + R(c^*, w_{\alpha_1})) \cdot \prod_{c\in\{C\setminus c^*\}} R(c, w_\alpha).$$

We can use left-distributivity to get

$$= \tfrac{\mathbf{1}}{\mathbf{2}} \cdot R(c^*, w_{\alpha_0}) \cdot \prod_{c\in\{C\setminus c^*\}} R(c, w_\alpha) + \tfrac{\mathbf{1}}{\mathbf{2}} \cdot R(c^*, w_{\alpha_1}) \cdot \prod_{c\in\{C\setminus c^*\}} R(c, w_\alpha).$$

At this point we know that $x_{i-1} \notin \mathcal{D}(c \in \{C \setminus c^*\})$, using Lemma 26 we get

$$\begin{aligned} &= \tfrac{\mathbf{1}}{\mathbf{2}} \cdot R(c^*, w_{\alpha_0}) \cdot \prod_{c\in\{C\setminus c^*\}} R(c, w_{\alpha_0}) + \tfrac{\mathbf{1}}{\mathbf{2}} \cdot R(c^*, w_{\alpha_1}) \cdot \prod_{c\in\{C\setminus c^*\}} R(c, w_{\alpha_1}) \\ &= \tfrac{\mathbf{1}}{\mathbf{2}} \cdot R(\bigwedge^p_{c\in C} c, w_{\alpha_0}) + \tfrac{\mathbf{1}}{\mathbf{2}} \cdot R(\bigwedge^p_{c\in C} c, w_{\alpha_1}) = \tfrac{\mathbf{1}}{\mathbf{2}} \cdot (R(\bigwedge^p_{c\in C} c, w_{\alpha_0}) + R(\bigwedge^p_{c\in C} c, w_{\alpha_1})). \end{aligned}$$

Partition Disjunction: Let the child nodes of $\vee^p$ be c, d.

$R(c \vee^p d, w_\alpha) = R(c, w_\alpha) + R(d, w_\alpha)$

Using the induction hypothesis on c, d we get

$$= \tfrac{\mathbf{1}}{\mathbf{2}}(R(c, w_{\alpha_0}) + R(c, w_{\alpha_1})) + \tfrac{\mathbf{1}}{\mathbf{2}}(R(d, w_{\alpha_0}) + R(d, w_{\alpha_1}))$$

We can use additive commutativity and distributivity to get

$$\begin{aligned} &= \tfrac{\mathbf{1}}{\mathbf{2}} \cdot (R(c, w_{\alpha_0}) + R(d, w_{\alpha_0}) + R(c, w_{\alpha_1}) + R(d, w_{\alpha_1})) \\ &= \tfrac{\mathbf{1}}{\mathbf{2}} \cdot (R(c \vee^p d, w_{\alpha_0}) + R(c \vee^p d, w_{\alpha_1})) \end{aligned}$$

Extended Frege can handle the bounded steps in each case of the inductive step. ◀

▶ **Lemma 28.** *For complete assignment α, we can prove using* eFrege *in the structure of the POG that $R(v, w_\alpha) = \mathbb{1}_v(\alpha)$.*

Proof. Again we show the base and inductive cases involve polynomially many basic steps.

Variable leaf: Let the variable leaf be x_i. $R(x_i, w_\alpha) = \mathbf{1}$ or $R(x_i, w_\alpha) = \mathbf{0}$ since $|\alpha| = |X|$ and the value is determined entirely by $\mathbb{1}_{x_i}(\alpha)$.

Negation: Suppose $R(c, w_\alpha) = \mathbf{1}$ then by the induction hypothesis α satisfies c, so α falsifies $\neg c$ and $R(\neg c, w_\alpha) = \mathbf{1} - R(c, w_\alpha) = \mathbf{0}$.
Similarly, it can be derived for $R(c, w_\alpha) = \mathbf{0}$ that $R(\neg c, w_\alpha) = \mathbf{1}$.

Partition Conjunction: Let C be the set of child nodes for $\wedge^p$. If there is some $c^* \in C$ such that α falsifies c^* then by the induction hypothesis $R(c^*, w_\alpha) = \mathbf{0}$. Then we can prove $\prod_{c\in C} R(c, w_\alpha) = \mathbf{0}$ which is the formula for $R(\bigwedge^p_{c\in C} c, w_\alpha)$. Also, α must falsify $\bigwedge^p_{c\in C} c$ as it falsifies c^*.
In the other case, if no $c \in C$ is falsified, α satisfies all of C (we can state and prove this formally as a disjunction). Here, $R(c, w_\alpha) = 1$ for all $c \in C$ and $\prod_{c\in C} R(c, w_\alpha) = \mathbf{1}$. Also, α must satisfy $\bigwedge^p_{c\in C} c$ as it satisfies all $c \in C$.

Partition Disjunction: Let the child nodes of $\vee^p$ be c, d. If α falsifies both c and d then by induction hypothesis, $R(c, w_\alpha) = R(d, w_\alpha) = \mathbf{0}$. By adding these we get $\mathbf{0} = R(c, w_\alpha) + R(d, w_\alpha) = R(c \vee^p d, w_\alpha)$. Also, α must falsify $c \vee^p d$ as it falsifies both c, d.
Suppose α satisfies c, we can prove that α falsifies d using the Resolution proof δ included in the CPOG proof (and this is simulated by eFrege). Hence, $R(c, w_\alpha) = \mathbf{1}$, $R(d, w_\alpha) = \mathbf{0}$. Then $R(c \vee^p d, w_\alpha) = \mathbf{1} + \mathbf{0} = \mathbf{1}$. This can be repeated for when α satisfies d by using left identity instead of right identity. ◀