The Relative Strength of #SAT Proof Systems[*]

Olaf Beyersdorff
Friedrich Schiller University Jena, Germany

Johannes K. Fichte
Linköping University, Sweden

Markus Hecher
Massachusetts Institute of Technology, Cambridge, MA, USA

Tim Hoffmann
Friedrich Schiller University Jena, Germany

Kaspar Kasche
Friedrich Schiller University Jena, Germany

Abstract

The propositional model counting problem #SAT asks to compute the number of satisfying assignments for a given propositional formula. Recently, three #SAT proof systems kcps (knowledge compilation proof system), MICE (model counting induction by claim extension), and CPOG (certified partitioned-operation graphs) have been introduced with the aim to model #SAT solving and enable proof logging for solvers.

Prior to this paper, the relations between these proof systems have been unclear and very few proof complexity results are known. We completely determine the simulation order of the three systems, establishing that CPOG simulates both MICE and kcps, while MICE and kcps are exponentially incomparable. This implies that CPOG is strictly stronger than the other two systems.

2012 ACM Subject Classification Theory of computation → Proof complexity; Theory of computation → Proof theory; Theory of computation → Logic and verification; Theory of computation → Automated reasoning

Keywords and phrases Model Counting, #SAT, Proof Complexity, Proof Systems, Lower Bounds, Knowledge Compilation

Digital Object Identifier 10.4230/LIPIcs.SAT.2024.5

Funding *Olaf Beyersdorff*: Carl-Zeiss Foundation and DFG grant BE 4209/3-1
Johannes K. Fichte: ELLIIT (Excellence Center at Linköping – Lund in Information Technology) funded by the Swedish government
Markus Hecher: Austrian Science Fund (FWF) grant J4656, Vienna Science and Technology Fund (WWTF) grant ICT19-065, and Society for Research Funding Lower Austria (GFF) grant ExzF-0004
Tim Hoffmann: Carl-Zeiss Foundation
Kaspar Kasche: Carl-Zeiss Foundation

Acknowledgements Parts of the work has been carried out while the first four authors were visiting the Simons Institute for Theory of Computation at UC Berkeley.

1 Introduction

The *propositional model counting problem* #SAT asks to compute the number of satisfying assignments for a given propositional formula [1]. The #SAT framework allows to efficiently encode and solve many real-world problems from areas such as probabilistic reasoning [5, 45],

[*] Authors are given in alphabetical order. Parts of the work has been carried out while the first four authors were visiting the Simons Institute for Theory of Computation at UC Berkeley.

risk analysis [26, 59] and explainable artificial intelligence [6, 51]. Interestingly, #SAT is among the hardest combinatorial problems and known to be #P-complete [5, 49, 55]. To put this in relation, by Toda's Theorem [53] any problem from the polynomial hierarchy (PH) can be solved in polynomial time by access to a #SAT oracle. In comparison, the SAT problem is on the first level of PH [20].

Over the last two decades, researchers and solver engineers improved effective *practical #SAT solving* [31] with numerous available #SAT solvers using conceptually quite different approaches. An annual competition captures current trends of solvers and novel practical algorithms, but also reveals that correctness needs to be improved [27].

In contrast to these practical advances, little is known theoretically on the power and limitations of #SAT solving. In both SAT and quantified Boolean formulas (QBF), the main theoretical approach towards gauging the strength of SAT and QBF solvers is through *proof systems and proof complexity* [13, 16]. The relation between proofs and solving is important in at least two aspects.

Firstly, proof systems can *model aspects of solving.* A seminal result in this direction is that CDCL solvers – the predominant approach in SAT solving – tightly correspond to propositional resolution [4, 7, 48], in the sense that any (non-deterministic) CDCL run on an unsatisfiable formula can be efficiently translated into a resolution refutation of the formula and vice versa. Further results are known for practical CDCL [56] and relations between QBF solving and related proof systems [10, 36]. This allows to apply the plethora of proof complexity results e.g. for propositional and QBF resolution [9, 43, 50] to the analysis of solvers. For example any lower bound for proof size in propositional resolution directly translates into a lower bound for CDCL runtime.

Secondly, proof systems can be employed for *proof logging and certifying solver correctness* by designing certified tools. Therefore, formal proof systems are introduced where a practical proof can be efficiently verified by a relatively simple method and easily emitted during solving. Different proof systems and formats have been designed including RUP, RAT and DRAT [29, 30, 34, 58] for SAT and QRAT [35] for QBF. These proof systems have been intensively studied and compared in terms of simulations, e.g., [19, 39, 40]. While the first modelling aspect needs weaker proof systems close to actual solving, the second proof-logging aspect favours very strong proof systems.

In comparison to the rich and intensely researched interplay between solving and proof complexity in SAT and QBF, significantly less is known in this regard for #SAT. In the past five years, *three different #SAT proof systems* have been introduced. These systems are kcps (2019) [18], MICE (2022) [11, 28], and CPOG (2023) [15]. These are the only #SAT proof systems so far.

The three proof systems are conceptually quite different: while kcps and CPOG are both static proof systems building on circuit classes used in knowledge compilation [24, 25] on which model counting is efficient, MICE is a rule-based proof system using three simple rules to compute counts for successively more complex formulas. The historically first system kcps was inspired by #SAT solving using knowledge compilation techniques. Both subsequent systems MICE and CPOG were designed with a view towards certifying different #SAT solving approaches.

In contrast to the rich proof complexity results for SAT and QBF, almost nothing is known theoretically for the three #SAT proof systems. Only for MICE, an exponential proof size lower bound was shown last year [11], while the relations between the three systems in terms of simulations are open.

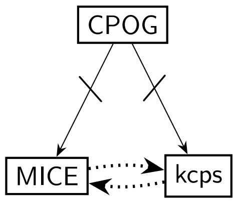


Figure 1 Simulation order of CPOG, MICE and kcps. A solid crossed edge from A to B indicates that A p-simulates B and A is exponentially stronger than B. Dotted lines indicate incomparability.

1.1 Our contributions

We perform a proof-complexity analysis of the three proof systems kcps, MICE and CPOG and completely determine their relative strength in terms of simulations and separations, leading to the picture in Figure 1. In more detail, our findings can be summarised as follows.

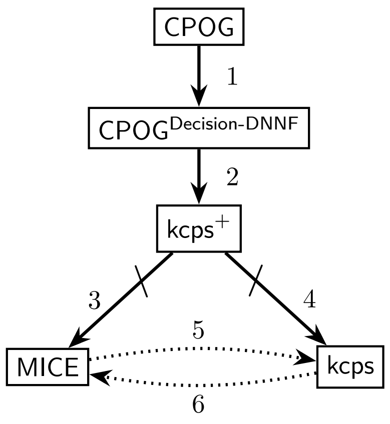
A. Simulations between #SAT proof systems. We formally compare the three #SAT proof systems in terms of *simulations* and show that CPOG p-simulates kcps and MICE, i.e. both kcps and MICE proofs can be efficiently translated into CPOG proofs.

Rather than showing the two simulations directly, we consider two intermediate proof systems kcps^{+} and $\mathsf{CPOG}^{\mathsf{Decision\text{-}DNNF}}$. The first of these was already suggested in [18] as a natural extension of kcps, while $\mathsf{CPOG}^{\mathsf{Decision\text{-}DNNF}}$ is newly introduced as a restriction of CPOG. The system CPOG uses POGs (partitioned-operation graphs) as the underlying circuit class, which in $\mathsf{CPOG}^{\mathsf{Decision\text{-}DNNF}}$ is restricted to Decision-DNNFs: the circuit class on which kcps and kcps^{+} are based. Representing a CNF by any of these circuit models allows efficient counting. Yet, the proofs need to contain additional information as verifying the equivalence of a CNF to a circuit in these models is non-trivial. While the two additional systems simplify our analysis, we believe they are also natural and of independent interest for further research (cf. the discussion in the conclusion).

For these five proof systems, we determine the simulation order as depicted in Figure 2, refining Figure 1 and including pointers to the results. While the simulations of kcps by kcps^{+} and of $\mathsf{CPOG}^{\mathsf{Decision\text{-}DNNF}}$ by CPOG follow almost by definition, the simulations of MICE by kcps^{+} and kcps^{+} by $\mathsf{CPOG}^{\mathsf{Decision\text{-}DNNF}}$ are more involved – in particular the first one – as they connect conceptually quite different proof formats. The proof systems kcps, kcps^{+}, $\mathsf{CPOG}^{\mathsf{Decision\text{-}DNNF}}$ and CPOG are all *static* as they are based on circuits equipped with additional information. In contrast, MICE is *rule-based* without any explicit connection to circuits.[1]

B. Exponential separations between #SAT proof systems. As our second main result we establish exponential separations between MICE and kcps in both directions. This entails exhibiting suitable CNF families that have short proofs in MICE, while requiring short kcps proofs, and vice versa. As a consequence, both systems are incomparable and at the same time exponentially weaker than kcps^{+} and CPOG, thus resulting in the situation depicted in Figure 1.

[1] However, it was noted already in [12] that from a MICE proof a Decision-DNNF for the CNF can be extracted. This does not, however, entail a simulation of MICE by kcps (which are based on Decision-DNNFs), and in fact such a simulation *fails* as implied by our separation results in B.



Edge	Simulation	Separation
1	Observation 4.5	open
2	Theorem 4.3*	open
3	Theorem 4.1	Corollary 5.14
4	Observation 4.2*	Corollary 5.14
5	(not possible)	Corollary 5.13
6	(not possible)	Corollary 5.4

Figure 2 Detailed simulation order of #SAT proof systems. A solid edge from A to B indicates that A p-simulates B. If the edge is crossed, A is also exponentially separated from B. A dotted edge from A to B indicates that A is exponentially separated from B. All the simulations of this paper require only logarithmic space; those highlighted with "$*$" only need linear time.

Technically, we obtain one direction (kcps does not simulate MICE) by showing a tight characterisation of kcps proof size by regular resolution size on unsatisfiable formulas. A similar characterisation of MICE by full resolution was shown in [11]. As regular resolution is known to be exponentially weaker than resolution [3, 57], the separation follows.

For the other direction (MICE does not simulate kcps) we use a variant of the pebbling formulas, prominent in propositional proof complexity [8, 14]. While the small Decision-DNNFs and short kcps proofs are relatively easy to obtain, the hardness argument for MICE is technically more involved (Theorem 5.7).

The first separation positively answers an open question posed by Capelli [18] to find CNFs with polynomial-size Decision-DNNFs (these can always be extracted from short MICE proofs [12]), but no small kcps proofs. The second separation implies that we cannot efficiently transform Decision-DNNFs into MICE proofs.

1.2 Related work

For decision proof systems, there are extensive studies on simulation and separation, see, e.g., [39, 40]. For recently defined proof systems for propositional model counting, this has been open. To the best of our knowledge, this paper is the first work in this direction for model counting proof systems. However, existing approaches have been studied empirically [15,18,28]. Indeed, there is a list [27] of practical exact model counting systems, which are based on different techniques. Among these are component caching [52], treewidth [42], knowledge compilation, e.g., d4 [44], c2d [24], dsharp [46], as well as hybrid approaches [33, 42]. Some theoretical results are presented in [17] which predates the introduction of formal proof systems for #SAT. There are also clausal proof systems enriched with XOR reasoning [47]. Very recently, first steps on proof systems for approximate counting [2] have been presented.

1.3 Organisation

The remainder of this paper is organised as follows. After reviewing some standard notions from propositional logic and proof systems in Section 2, we provide formal definitions of the existing proof systems for #SAT in Section 3. We show the simulations from Figure 2 in Section 4. The separations are provided in Section 5. We conclude in Section 6 with a discussion on practical and theoretical implications.

We highlight that though we believe our results bear practical relevance, this paper performs a purely theoretical proof-complexity investigation.

2 Preliminaries

We briefly provide formal notions from propositional logic and proof systems. For more detailed information, we refer to [1, 41]. For an integer n, we set $[n] := \{1, 2, \ldots, n\}$.

Propositional formulas. A *literal* l is a variable z or its negation $\overline{z}$, and we write $\mathsf{var}(l) := z$. A *clause* is a disjunction of literals, a *conjunctive normal form (CNF)* is a conjunction of clauses. Often, we write clauses as sets of literals and formulas as sets of clauses. We assume that *propositional formulas* are given in CNF. We can efficiently transform any formula into a CNF using Tseitin transformations [54]. For a formula F, $\mathsf{vars}(F)$ denotes the set of all variables in F. If $C \in F$ is a clause and $V \subseteq \mathsf{vars}(F)$ is a set of variables, we define $C|_V = \{l \in C \mid \mathsf{var}(l) \in V\}$ and $F|_V$ denotes the formula F with every clause C replaced by $C|_V$.

Given a set V of variables, a (partial) assignment is a (partial) function $\alpha : V \to \{0, 1\}$ that maps variables to Boolean values. We write $\langle V \rangle$ for the set of all $2^{|V|}$ complete assignments to V. For a (partial) assignment α, $F[\alpha]$ denotes the formula where we replace all occurrences of variables x with $\alpha(x)$. If $F[\alpha] = 1$, we say α *satisfies* F and write $\alpha \models F$. We say that α *falsifies* F if $F[\alpha] = 0$ and write $\alpha \not\models F$.

Occasionally, we interpret an assignment as a CNF consisting of precisely those unit clauses that specify the assignment. Therefore, the set operations are well defined for formulas and assignments. We say that two assignments are *consistent* if they agree on their intersection.

A formula F is *satisfiable* if there exists an assignment $\alpha \in \langle \mathsf{vars}(F) \rangle$ such that $\alpha \models F$ and is *unsatisfiable* if there exists no such assignment. For a formula φ, $\mathsf{Mod}(\varphi) := \{\alpha \in \langle \mathsf{vars}(\varphi) \rangle \mid \alpha \models \varphi\}$ is the set of all models of φ. The *model counting problem* #SAT asks to compute $|\mathsf{Mod}(\varphi)|$ for a given formula φ. Throughout the paper, we use φ for formulas we want to count on. The SAT problem asks to decide whether a given formula is satisfiable and UNSAT whether a given formula is unsatisfiable. Moreover, we need the definition of *semantic consequence.* We write $F \models G$ if and only if for every assignment $\alpha \in \langle \mathsf{vars}(F) \rangle$, we have that $\alpha \models F$ implies $\alpha \models G$. We write $F \equiv G$ if and only if $F \models G$ and $G \models F$.

Proof systems. Following Cook and Reckhow [21], a *proof system* for a language L is a polynomial-time computable function f with range $\mathsf{rng}(f) = L$. Here, L will be chosen as either UNSAT or #SAT. If $f(w) = x$, then w is called f-proof of $x \in L$. In order to compare proof systems we need the notion of *simulations.* Let P and Q be proof systems for the same language. Then, P *p-simulates* Q if every Q-proof can be translated in polynomial time into a P-proof of the same formula. Two proof systems are *p-equivalent* if they p-simulate each other. Further, P is *exponentially separated* from Q if there is a family of formulas that have polynomial sized P-proofs while any Q-proof requires exponential size.

Resolution is arguably the most studied proof system for UNSAT. It is a *line-based* proof system with clauses as proof lines. The *resolution rule* allows to derive the clause $C \cup D$ from previously derived clauses $C \cup \{x\}$ and $D \cup \{\overline{x}\}$. We also allow the *weakening rule* that derives $C \cup D$ from a clause C. A resolution refutation of a CNF is a derivation of the empty clause $\square$. As refutational systems, resolution with and without weakening a p-equivalent.

Further, we can interpret any proof π in a line-based proof system as a directed graph G_π, where the nodes are proof lines in π. There is an edge from proof line l_1 to l_2 if l_2 was used to derive l_1. A resolution refutation is *regular* if there is no path from the root to a leaf in G_π where a variable is resolved more than once.

3 Proof systems for #SAT

In this section, we recall the existing #SAT proof systems kcps, CPOG, and MICE and provide some intuition. In particular, we provide a concise formalisation of CPOG. Furthermore, we introduce two adapted versions of kcps and CPOG that we call kcps^+ and $\mathsf{CPOG}^{\mathsf{Decision\text{-}DNNF}}$. As kcps and CPOG heavily use concepts from knowledge compilation, we start with relevant definitions following standard texts [25, 37].

A *circuit* is a directed acyclic graph with labelled nodes that we call *gates*. We only consider circuits that have exactly one gate with indegree 0. It is called *root* and represents the circuit's output. Gates with outdegree 0 are called *leaves* and are labelled with literals or constants 0 and 1. The latter are also called 0-*gate* or 1-*gate*. Every inner gate is an AND-, OR- or NOT-gate labelled with the corresponding Boolean function. The semantics of such circuits are defined in the usual way. Additionally, we assume that AND- and OR-gates always have exactly two children.

Let D be a circuit. For gates in D we use uppercase letters such as N. We write $\mathsf{vars}(D)$ for the set of all variables occurring in leaves of D. $\mathcal{E}(D)$ denotes the Tseitin encoding [54] of D, where we use a new variable ϑ_N for every gate N. We denote the subcircuit of D with root N consisting of all descendants of N by $D(N)$.

A circuit is in *negation normal form (NNF)* if it does not contain NOT-gates. An AND-gate with children N_1 and N_2 is called *decomposable*, if $\mathsf{vars}(D(N_1)) \cap \mathsf{vars}(D(N_2)) = \emptyset$. An OR-gate with children N_1 and N_2 is called *deterministic* if there is no assignment that satisfies both $D(N_1)$ and $D(N_2)$. A DNNF [22] is an NNF where every AND-gate is decomposable. A d-DNNF [23] is a DNNF where every OR-gate is deterministic.

Since it is non-trivial to check if all OR-gates are indeed deterministic, we also consider a restricted version of d-DNNF called Decision-DNNF. In a Decision-DNNF, any OR-gate has the form $N = (N_1 \texttt{ or } N_2)$ with $N_1 = (x \texttt{ and } N_3)$ and $N_2 = (\overline{x} \texttt{ and } N_4)$ for any variable x. It is obvious that any such OR-gate is deterministic. For better readability, we write Decision-DNNFs without OR-gates but use DECISION-gates instead. We rewrite the above gate as $N = (\texttt{if } x \texttt{ then } N_3 \texttt{ else } N_4)$. Note that we can assume that the leaves of a Decision-DNNF contain only constants 0 or 1. Further, in any path from the root to a leaf, any variable can be decided at most once because of the decomposability property. We say that an assignment α *reaches* a gate N if there is a path P from the root to N such that all decisions along P are consistent with α.

3.1 Kcps: Knowledge Compilation Proof System

The system kcps is the historically first proof system for #SAT, introduced by Capelli in 2019 [18]. As the name suggests, it aims to certify solvers that apply knowledge compilation techniques. These solvers transform the input CNF into a format that can handle various queries efficiently, in particular model counting [24, 25, 44]. As in practice solvers often rely on compiling the formulas into Decision-DNNFs, kcps is based on this class of circuits.

A kcps proof of φ provides a Decision-DNNF D such that $D \equiv \varphi$. The Decision-DNNF D implicitly contains the model count of φ as we can efficiently compute it (cf. Figure 3 for an example). However, for this to be a proof in the sense of Cook-Reckhow [21], we need

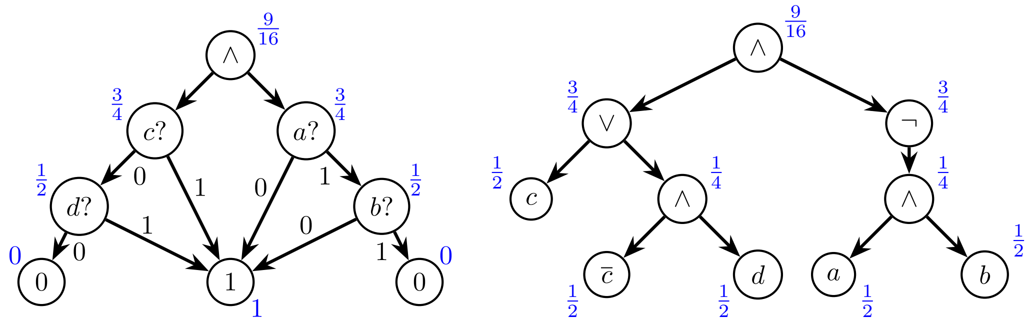


Figure 3 A Decision-DNNF (left) and a POG (right) that are equivalent to the formula $\varphi = (\overline{a} \vee \overline{b}) \wedge (c \vee d)$. The blue number at a gate N indicates the fraction of assignments that satisfy the subcircuit with root N. These numbers are computed bottom-up, i.e. 0-gates get count 0, 1-gates count 1 and gates that are labelled with a literal count $\frac{1}{2}$. DECISION-gates get the average of the two children, OR-gates the sum and AND-gates the product. Finally, to compute the model count of φ, we multiply the fraction of satisfying assignments of the whole Decision-DNNF or POG with the count of all possible assignments to $\mathsf{vars}(\varphi)$, resulting in $|\mathsf{Mod}(\varphi)| = \frac{9}{16} \cdot 2^{|\mathsf{vars}(\varphi)|} = 9$.

to verify that D and φ are indeed equivalent. The direction $D \models \varphi$ can always be checked efficiently [25] (cf. also Lemma 3.6 below for a formal argument). However, the other direction $\varphi \models D$ is a coNP-complete problem for arbitrary Decision-DNNFs [18]. Thus, we consider a restricted version of Decision-DNNFs on which checking $\varphi \models D$ becomes easy as well. For that, we review the notion of a certified Decision-DNNF [18].

▶ **Definition 3.1** (S-certified Decision-DNNF [18]). *Let S be a set of clauses. A Decision-DNNF D is called S-*certified *if every 0-gate N is labelled by a certificate $C \in S$. A clause is a certificate for N if all assignments that reach N falsify C.*

These restricted Decision-DNNFs have the property that for a formula φ, any φ-certified Decision-DNNF D satisfies $\varphi \models D$ [18]. To see this, consider the equivalent statement $\neg D \models \neg\varphi$. Let α be an assignment that falsifies D, then it reaches a 0-gate. Consequently, it has to falsify its certificate and in particular φ.

Finally, we can define the kcps proof system:

▶ **Definition 3.2** (kcps [18]). *A* kcps *proof of a CNF φ is a φ-certified Decision-DNNF D where $D \equiv \varphi$.*

Note that the model count of φ and also the equivalence between φ and D are not explicitly part of the proof as we can compute the model count efficiently from D and verify $D \equiv \varphi$ in polynomial time.

In fact, Capelli [18] proposed a generalization of kcps where the certifying clauses for the 0-gates are not necessarily clauses of the original formula φ. Instead, we use as certificates arbitrary clauses derived by resolution from φ. This results in the proof system kcps$^+$.

▶ **Definition 3.3** (kcps$^+$ [18]). *A* kcps$^+$ *proof of a CNF φ is a pair (σ, D) where*

1. *σ is a resolution derivation starting from the clauses in φ and*
2. *D is a σ-certified Decision-DNNF (i.e. all clauses labelling the 0-gates in D are derived in σ) such that $D \equiv \varphi$.*

3.2 CPOG: Certified Partitioned-Operation Graphs

In contrast to kcps, CPOG is not restricted to certified Decision-DNNFs, but uses the more flexible circuit class POG (partitioned-operation graphs). Instead of providing the original definition of POGs from [15], we equivalently define a POG as a d-DNNF with NOT-gates (alternatively, a d-DNNF can be viewed as a POG with negation applied only to variables).

Model counting is also efficient on POGs [15], and in fact POGs appear to be the largest class to which the model counting idea used for Decision-DNNFs naturally extends. However, in order to maintain efficient proof checking, a CPOG proof has to explicitly prove that P is indeed a POG and that $\varphi \equiv P$. This leads to the following definition.

▶ **Definition 3.4** (CPOG [15]). *A* CPOG *proof of a CNF φ is a 4-tuple $(\mathcal{E}(P), \rho, \psi, X)$ where*

1. *P is a* POG *with root R such that $P \equiv \varphi$ and $\mathcal{E}(P)$ is a clausal encoding of P,*
2. *ρ is a proof for $\varphi \models P$, i.e., ρ is a resolution refutation of $\mathcal{E}(P) \wedge \varphi \wedge (\overline{\vartheta_R})$,*
3. *ψ is a proof for $P \models \varphi$, i.e., ψ contains a resolution refutation of $\mathcal{E}(P) \wedge (\vartheta_R) \wedge \overline{C}$ for every clause $C \in \varphi$,*
4. *X is a set of proofs verifying that all OR-gates of P are deterministic, i.e., X is a set of resolution refutations such that for any OR-gate N, X contains a resolution refutation of $\mathcal{E}(P) \wedge (\vartheta_{N_1}) \wedge (\vartheta_{N_2})$, where N_1 and N_2 are the two child gates of N.*

Note that CPOG is originally defined on circuits with arbitrary fan-in, however we consider only the binary case which is polynomially equivalent. Additionally, the original definition uses RUP steps for the propositional proofs, which are p-equivalent to resolution. In our definition, we use resolution proofs instead.

The underlying structure of POGs in CPOG proofs is quite generic. So far, the only implementation of CPOG [15] is restricted to Decision-DNNFs instead of POGs. We capture this variant in the following definition:

▶ **Definition 3.5** ($\mathsf{CPOG}^{\text{Decision-DNNF}}$). *A* $\mathsf{CPOG}^{\text{Decision-DNNF}}$ *proof of a CNF φ is a pair $(\mathcal{E}(D), \rho)$ where*

1. *D is a Decision-DNNF with root R and $\mathcal{E}(D)$ a clausal encoding of D such that $D \equiv \varphi$,*
2. *ρ is a resolution refutation of $\varphi \wedge \mathcal{E}(D) \wedge (\overline{\vartheta_R})$.*

Note that in comparison to Definition 3.4, the last two items are missing. This is clear for item 4 as D only contains DECISION-gates instead of OR-gates. But also the proof ψ in item 3 can always be computed efficiently for Decision-DNNFs as we show in the next lemma.

▶ **Lemma 3.6.** *Let D be a Decision-DNNF with root R and encoding $\mathcal{E}(D)$. If $D \equiv \varphi$, then we can compute ψ, i.e. a resolution refutation of $\mathcal{E}(D) \wedge (\vartheta_R) \wedge \overline{C}$ for every clause $C \in \varphi$, in time $O(|D| \cdot |\varphi|)$.*

Proof sketch. Let $C \in \varphi$ be some arbitrary fixed clause. We assume that $D \equiv \varphi$. Let $L = N_1, \dots, N_n$ be a list of all gates in D that are unsatisfiable under the partial assignment $\overline{C}$ in some topological ordering such that no gate is listed after its ancestors. Note that R is the last element in L as $D[\overline{C}] \equiv \varphi[\overline{C}]$ has to be unsatisfiable. We can show inductively that for every $i \in [n]$ we can effectively derive the unit clause $(\overline{\vartheta_{N_i}})$ from $\mathcal{E}(D) \wedge \overline{C}$. We do this by deriving $(\overline{\vartheta_{N_i}})$ from the corresponding unit clauses of its children in a constant number of resolution steps.

Since $R \in L$, we can derive $(\overline{\vartheta_R})$ efficiently. With an additional resolution step with the unit clause (ϑ_R), we derive the empty clause. In total, we can construct a resolution refutation of $\mathcal{E}(D) \wedge (\vartheta_R) \wedge \overline{C}$ of size $O(|D|)$. Since ψ contains such a refutation for every clause $C \in \varphi$, we can construct ψ with at most $O(|D| \cdot |\varphi|)$ resolution steps. ◀

Axiom. $\dfrac{}{(\emptyset, \emptyset, 1)}$ (Ax)

Composition. $\dfrac{(F, A_1, c_1) \quad \cdots \quad (F, A_n, c_n)}{(F, A, \sum_{i\in[n]} c_i)}$ (Comp)

(C-1) $\mathsf{vars}(A_1) = \cdots = \mathsf{vars}(A_n)$ and $A_i \neq A_j$ for $i \neq j$,
(C-2) $A \subseteq A_i$ for all $i \in [n]$
(C-3) there exists a resolution refutation of $A \cup F \cup \{\overline{A}_i \mid i \in [n]\}$. Such a refutation is included into the trace and is called an *absence of models statement.*

Join. $\dfrac{(F_1, A_1, c_1) \quad (F_2, A_2, c_2)}{(F_1 \cup F_2, A_1 \cup A_2, c_1 \cdot c_2)}$ (Join)

(J-1) A_1 and A_2 are consistent,
(J-2) $\mathsf{vars}(F_1) \cap \mathsf{vars}(F_2) \subseteq \mathsf{vars}(A_i)$ for $i \in \{1, 2\}$.

Extension. $\dfrac{(F_1, A_1, c_1)}{(F, A, c_1 \cdot 2^{|\mathsf{vars}(F) \setminus (\mathsf{vars}(F_1) \cup \mathsf{vars}(A))|})}$ (Ext)

(E-1) $F_1 \subseteq F$,
(E-2) $A|_{\mathsf{vars}(F_1)} = A_1$,
(E-3) A satisfies $F \setminus F_1$.

Figure 4 Inference rules for MICE [11].

3.3 MICE: Model-counting Induction by Claim Extension

The third system we need is the line-based #SAT proof system MICE, introduced with the intention to provide a proof system close to various solvers [28]. Here, we use MICE in its slightly simplified, but p-equivalent form as defined in [11].

▶ **Definition 3.7** (MICE [11, 28]). *The proof lines in* MICE *are called* claims*. A* claim *is a* 3*-tuple* (F, A, c) *consisting of a CNF* F*, a partial assignment* A *of* $\mathsf{vars}(F)$ *(called* assumption*) and a count* c*. A* MICE proof *of a CNF* φ *is a sequence of claims* $I_1, \ldots, I_k$ *that are derived with the inference rules in Figure 4 such that the final claim has the form* $(\varphi, \emptyset, c)$ *for some count* c*.*

If a MICE proof π derives the claim $(\varphi, \emptyset, c)$, then π proves that φ has exactly c models. A claim (F, A, c) is *correct* if F has exactly c models that are consistent with A. Since only correct claims can be derived in MICE [11], the count c of a correct claim (F, A, c) is uniquely determined by F and A. Thus, we sometimes omit c and use the notation (F, A) instead.

4 CPOG simulates MICE and kcps

We start our investigation by clarifying the simulation order of the #SAT proof systems introduced in Section 3 and prove that CPOG simulates MICE and kcps. We achieve this by efficiently constructing CPOG proofs from given MICE or kcps proofs. We use the systems $\mathsf{CPOG}^{\mathsf{Decision\text{-}DNNF}}$ and kcps^{+} from Section 3 as convenient intermediate proof systems and show the four simulations depicted in Figure 2.

Our first simulation transforms MICE proofs into kcps^{+} proofs.

▶ **Theorem 4.1.** kcps^{+} *p-simulates* MICE*.*

Proof sketch. Let $\pi = I_1, \ldots, I_n$ be a MICE proof of a CNF φ with $I_k = (F_k, A_k)$ for every $k \in [n]$. Our goal is to construct a kcps^+ proof for φ from π. W.l.o.g. the first claim of π is $(\emptyset, \emptyset, 1)$ derived with (Ax) and all other claims are not derived with (Ax). For each $k \in [n]$, we construct a Decision-DNNF D_k that satisfies the following invariants:

(i) D_k is equivalent to $F_k[A_k]$,
(ii) D_k contains only variables from $F_k[A_k]$, and
(iii) every 0-gate N in D_k is labelled with some clause C derived from φ such that for any assignment $\alpha \in \langle \mathsf{vars}(D_k) \rangle$ that reaches N, the clause C is falsified by $\alpha \cup A_k$.

For the *base case* $k = 1$, $I_1 = (\emptyset, \emptyset, 1)$ is derived with (Ax). We set D_1 to a circuit that only contains one 1-gate. For the *induction step*, we distinguish how I_k is derived.

- *Join.* I_k is derived with (Join) from claims I_i and I_j, so we have $F_k = F_i \cup F_j$ and $A_k = A_i \cup A_j$. Per induction hypothesis, we have already derived Decision-DNNFs D_i and D_j equivalent to $F_i[A_i]$ and $F_j[A_j]$. We define D_k to be an AND-gate with the two children that are the roots of D_i and D_j.
- *Composition.* I_k is derived with (Comp) from claims $I_{i_1}, \ldots, I_{i_r}$. Per induction hypothesis, we have the corresponding circuits D_{i_j} for all $j \in [r]$. Let $V = \mathsf{vars}(A_{i_1}) \setminus \mathsf{vars}(A_k)$, keeping in mind that all assumptions A_{i_j} have the same set of variables because of (C-1). We build a complete binary decision tree T with variables in V. For every claim I_{i_j} for $j \in [r]$ there is exactly one leaf in T that is consistent with the assumption A_{i_j}. We replace this leaf with the root of the corresponding Decision-DNNF D_{i_j}. Afterwards, we replace all remaining leaves with a 0-gate. Furthermore, we remove every DECISION-gate where both decisions lead to a 0-gate as long as such gates exist. We set D_k to be the resulting circuit. For each new 0-gate we can specify a valid certificate and construct its derivation from the absence of models statement that was used for the (Comp).
- *Extension.* I_k is derived with (Ext) from I_i. Per induction hypothesis, we have already derived a Decision-DNNF D_i equivalent to $F_i[A_i]$. We set $D_k = D_i$.

This completes the induction. Since $I_n = (\varphi, \emptyset)$, D_n is a Decision-DNNF representing φ. Further, all 0-gates have valid certificates in some derivation σ. Therefore, we have constructed a valid kcps^+ proof $\pi' = (D_n, \sigma)$. With $|D_n| = O(n^2 \cdot |\mathsf{vars}(\varphi)|)$ and $|\sigma| = O(n^2 \cdot |\mathsf{vars}(\varphi)| \cdot |\pi|)$, we get that $|\pi'|$ is polynomial in $|\pi|$. ◀

We remark that there is a related result in [12], which shows that we can efficiently transform any MICE proof of some formula φ into a Decision-DNNF D representing φ. The theorem above strengthens this by showing that we can even derive some set σ of clauses such that all 0-gates of D are σ-certified.

Next, we observe that kcps^+ is indeed a generalization of kcps. This holds as we can write any kcps proof D as a kcps^+ proof (σ, D) where σ contains all clauses of φ.

▶ **Observation 4.2.** kcps^+ *p-simulates* kcps.

Now, we efficiently transform a given kcps^+ proof of a CNF φ into a $\mathsf{CPOG}^{\mathsf{Decision\text{-}DNNF}}$ proof. The choice of the Decision-DNNF D for the CPOG proof is obvious: we simply copy it from the kcps^+ proof. Therefore, we only have to construct a short refutation of $\varphi \models D$.

▶ **Theorem 4.3.** $\mathsf{CPOG}^{\mathsf{Decision\text{-}DNNF}}$ *p-simulates* kcps^+.

Proof. Let $\pi = (\sigma, D)$ be a kcps^+ proof for φ. Further, let $\mathcal{E}(D)$ be the clausal Tseitin encoding of the Decision-DNNF D with root R. For any resolution refutation ρ of $\varphi \wedge \mathcal{E}(D) \wedge (\overline{\vartheta_R})$, we obtain a valid $\mathsf{CPOG}^{\mathsf{Decision\text{-}DNNF}}$ proof $\pi' = (\mathcal{E}(D), \rho)$. In order to prove the theorem, we construct ρ such that $|\pi'| = O(|\pi|)$. As $|\pi'| = |\mathcal{E}(D)| + |\rho| = O(|D|) + |\rho|$ it is sufficient that $|\rho| = O(|D| + |\sigma|)$. For that, we first derive all clauses of σ in ρ.

▷ Claim 4.4. For every gate N in D, we can efficiently derive a clause $C_N = (\vartheta_N \vee C)$ from $\mathcal{E}(D) \wedge \varphi$ where C is a clause satisfying the invariant (I): *Any assignment leading to N falsifies C.*

Proof sketch. We show this by induction on the gates of D starting at the leaves. In the base case, N is a leaf. If N is a 0-gate with certificate C', we can derive C_N for $C = C'$. Otherwise, if N is an 1-gate, we can derive C_N with $C = \emptyset$. In the induction step, we use the already derived clauses C_1, C_2 corresponding to the children of N. Together with $\mathcal{E}(D)$ we can derive the clause C_N with a constant number of resolution steps. ◁

As a result, we can also derive the clause $C_R = (\vartheta_R \vee C)$ for the root R such that C satisfies (I). As there are no decisions above R, C has to be the empty clause, i.e., we have derived the unit clause $C_R = (\vartheta_R)$. By applying a resolution step with the other unit clause $(\overline{\vartheta_R})$, we refute $\varphi \wedge \mathcal{E}(D) \wedge (\overline{\vartheta_R})$ with resolution.

In total, ρ contains the derivations of σ and additionally, a constant number of resolution steps for each gate in D. Thus, the resulting resolution refutation ρ has at most size $|\sigma| + O(|D|)$ and the theorem follows. ◀

We finally show last simulation which almost follows by definition as POGs generalise Decision-DNNFs.

▶ **Observation 4.5.** CPOG *p-simulates* CPOG$^{\text{Decision-DNNF}}$.

Proof sketch. In order to transform a CPOG$^{\text{Decision-DNNF}}$ proof $(\mathcal{E}(D), \rho)$ into a CPOG proof $(\mathcal{E}(P), \rho, \psi, X)$, we use that D is also a POG by setting $P = D$. We can compute the corresponding ψ efficiently as shown in Lemma 3.6. Further, it is easy to show with resolution that all Or-gates are deterministic, i.e., X can also be computed efficiently. ◀

With that, we have shown all simulations illustrated in Figure 2. Upon closer examination, all these simulations turn out to be computable with logarithmic space. Moreover, the simulations in Observation 4.2 and Theorem 4.3 can be computed in linear time.

5 MICE and kcps are incomparable

Having determined the simulation order of #SAT proof systems, we now turn to lower bounds and separations between them. We first compare MICE and kcps.

5.1 CNFs that are hard for kcps but easy for MICE

Before getting to specific lower bounds, we provide a tight characterisation of proof size on unsatisfiable formulas for kcps in terms of regular resolution.

▶ **Proposition 5.1.** *For unsatisfiable formulas,* kcps *and regular resolution are p-equivalent.*

Proof. The proof is based on [37, Theorem 18.1] stating that the minimal size of any regular resolution refutation of a formula φ equals the minimal size of any read-once branching program solving the search problem for φ. A read-once branching program for the search problem for φ is equivalent to a φ-certified Decision-DNNF D with $D \equiv \varphi$ that contains no And-gates. Thus, the result directly implies that kcps p-simulates regular resolution for unsatisfiable formulas.

For the converse simulation of kcps by regular resolution we consider an arbitrary φ-certified Decision-DNNF D with $D \equiv \varphi$ for some unsatisfiable formula φ and show that we can get rid of all And-gates:

▷ Claim 5.2. There is a φ-certified Decision-DNNF D' with $D' \equiv \varphi$ and $|D'| \leq |D|$ that contains no AND-gates.

Proof sketch. To prove this claim we present a technique to remove an AND-gate. Let N be an AND-gate of D such that all ancestors of N are DECISION-gates. Further, let N_1 and N_2 be the children of N, i.e. $D(N) \equiv D(N_1) \wedge D(N_2)$. We can argue that not both $D(N_1)$ and $D(N_2)$ are satisfiable as this would lead to a satisfying assignment of φ.

Therefore, we can assume w.l.o.g. that $D(N_1)$ is unsatisfiable, i.e. $D(N) \equiv D(N_1) \wedge D(N_2) \equiv D(N_1)$. Thus, we can decrease the number of AND-gates of D by 1 by replacing gate N with N_1. This can never increase the set of assignments that reach a particular gate, and therefore leaves all certificates intact. In this way, we can remove every AND-gate one by one without increasing the size of D. ◁

By using this claim, we convert D to some D' without AND-gates, apply the result from [37, Theorem 18.1] and obtain a regular resolution refutation of size at most $|D|$. ◀

Therefore, any lower (and upper) bound for regular resolution transfers to kcps. For regular resolution, many lower bounds are known [43], and in particular all formulas hard for resolution such as the pigeonhole principle [32] are hard for kcps. Note that any unsatisfiable formula has a trivial Decision-DNNF. Nevertheless, all kcps proofs can be of exponential size. This answers an open question from [18].

A similar proof size characterisation on unsatisfiable formulas is known for MICE, in this case in terms of full (i.e. unrestricted) resolution [11].

▶ **Proposition 5.3** ([11]). *For unsatisfiable formulas,* MICE *and resolution are p-equivalent.*

As there are CNF families exponentially separating regular and full resolution [3, 57], Propositions 5.1 and 5.3 yield:

▶ **Corollary 5.4.** MICE *is exponentially separated from* kcps.

While this separation is on unsatisfiable formulas, we can easily extend it to satisfiable CNFs as well. For this, let φ be an unsatisfiable formula that separates resolution from regular resolution. For some fresh variable $a \notin \mathsf{vars}(\varphi)$, we define $\varphi' = \{(C \vee a) \mid C \in \varphi\}$. Then, φ' has $2^{|\mathsf{vars}(\varphi)|}$ models and still separates MICE from kcps.

Firstly, φ' is still easy for MICE. We derive the claim $(\varphi', \{\overline{a}\})$ with (Comp), the absence of models statement is short since $\varphi'[\overline{a}] = \varphi$ has a short resolution refutation. Further, we derive $(\varphi', \{a\})$ with (Ext) and finally apply (Comp) to these two claims, resulting in $(\varphi', \emptyset)$.

Secondly, we argue that the hardness of φ for kcps implies the hardness of φ'. For the contrapositive, let $D \equiv \varphi'$ be a φ'-certified Decision-DNNF. Then, $D[\overline{a}] \equiv \varphi$ is a φ-certified Decision-DNNF of size at most $|D|$, i.e. φ has a kcps proof of analogous size.

5.2 CNFs that are hard for MICE but easy for kcps

Next, we show that MICE cannot simulate kcps. For that, we use a variant of the pebbling formulas on pyramidal graphs. For a given size $n \in \mathbb{N}$, the pyramidal graph (cf. Figure 5) has $m := \frac{n(n+1)}{2}$ nodes: a node $P_{i,j}$ for each $1 \leq j \leq i \leq n$. For each $i < n$, there are edges from $P_{i+1,j}$ and $P_{i+1,j+1}$ to $P_{i,j}$. The variable i is called the *row* of the node, and j is called the *column*. When comparing rows, we talk about *greater* or *smaller* rows. The nodes in row n are called sources, and the node in row 1 is called the sink.

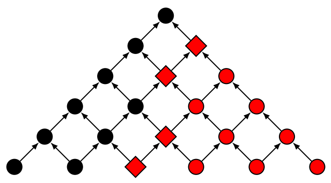


Figure 5 Pyramidal graph for PEB_6, depicting the situation in the proof of Theorem 5.7. For a fixed claim in a MICE proof of PEB_6, the red nodes are *active*, i.e. they correspond to variables that occur in the formula F. The diamond-shaped nodes form the *boundary* of this claim as they have neighbours that are not active.

We start with some intuition for the pebbling formulas PEB_n. They have two variables $w_{i,j}$ and $b_{i,j}$ for each node $P_{i,j}$. $w_{i,j}$ represents a white pebble being placed on that node, while $b_{i,j}$ represents a black pebble. The formula requires each source node to contain a pebble. Every other node needs to contain a pebble if and only if both its parent nodes contain a pebble. No node can simultaneously contain a black and a white pebble.

▶ **Definition 5.5.** *Let n be an integer. The formula PEB_n has variables $w_{i,j}$ and $b_{i,j}$ for every $i, j \in [n]$ with $j \le i$. PEB_n is a CNF defined as follows:*

- *For every $i, j \in [n-1], j \le i$ the formula requires that*

$$(w_{i,j} \vee b_{i,j}) \leftrightarrow ((w_{i+1,j} \vee b_{i+1,j}) \wedge (w_{i+1,j+1} \vee b_{i+1,j+1})).$$

  This is expressed using the clauses

$$C^1_{i,j} = \overline{w_{i+1,j}} \vee \overline{w_{i+1,j+1}} \vee w_{i,j} \vee b_{i,j} \qquad C^2_{i,j} = \overline{w_{i+1,j}} \vee \overline{b_{i+1,j+1}} \vee w_{i,j} \vee b_{i,j}$$
$$C^3_{i,j} = \overline{b_{i+1,j}} \vee \overline{w_{i+1,j+1}} \vee w_{i,j} \vee b_{i,j} \qquad C^4_{i,j} = \overline{b_{i+1,j}} \vee \overline{b_{i+1,j+1}} \vee w_{i,j} \vee b_{i,j}$$
$$C^5_{i,j} = w_{i+1,j} \vee b_{i+1,j} \vee \overline{w_{i,j}} \qquad C^6_{i,j} = w_{i+1,j} \vee b_{i+1,j} \vee \overline{b_{i,j}}$$
$$C^7_{i,j} = w_{i+1,j+1} \vee b_{i+1,j+1} \vee \overline{w_{i,j}} \qquad C^8_{i,j} = w_{i+1,j+1} \vee b_{i+1,j+1} \vee \overline{b_{i,j}}.$$

- *For every $j \in [n]$ there is a clause $w_{nj} \vee b_{nj}$.*
- *For every $i, j \in [n], j \le i$ there is a clause $C^9_{i,j} = \overline{b_{i,j}} \vee \overline{w_{i,j}}$.*

Note that the commonly used pebbling formulas require the sink node $P_{1,1}$ to contain no pebbles, making the formula unsatisfiable. We omit this requirement and obtain a formula that is satisfied if and only if each node contains exactly one pebble. It has 2^m models where m is the number of nodes. Two nodes are called *adjacent* if there is an edge between them in the pebbling graph.

To separate kcps from MICE with PEB_n, we show that there are polynomial-sized proofs in kcps while any MICE proof requires exponential size. We start with the upper bound.

▶ **Proposition 5.6.** *There is a* kcps *proof of PEB_n of size $O(|\mathsf{PEB}_n|)$.*

Proof. We iteratively construct a PEB_n-certified Decision-DNNF D with $D \equiv \mathsf{PEB}_n$. For each node $P_{i,j}$, we construct a partial Decision-DNNF with root $N_{i,j}$ that handles the case $\{w_{i,j} = 0, b_{i,j} = 0\}$. This means that in order to obtain a valid Decision-DNNF, all paths to $N_{i,j}$ must include these two decisions. We also make sure that descendants of $N_{i,j}$ only decide variables of nodes in rows greater than i.

We begin constructing the $N_{i,j}$ for greater rows, starting with $i = n$, and continue to smaller rows. For $i = n$, $N_{i,j}$ is simply a 0-gate labelled with the clause $w_{n,j} \vee b_{n,j}$, which will be falsified by the assumption $\{w_{i,j} = 0, b_{i,j} = 0\}$.

$$\begin{tikzcd}
 & & 0\ (C^1_{i,j}) & & \\
 & w_{i+1,j+1}? \arrow[ur, "1"] \arrow[dr, "0"'] & & 0\ (C^2_{i,j}) & \\
 & & b_{i+1,j+1}? \arrow[ur, "1"] \arrow[dr, "0"'] & & \\
N_{i,j} : w_{i+1,j}? \arrow[uur, "1"] \arrow[dddr, "0"'] & & & (N_{i+1,j+1}) & \\
 & & & 0\ (C^3_{i,j}) & \\
 & & w_{i+1,j+1}? \arrow[ur, "1"] \arrow[dr, "0"'] & & 0\ (C^4_{i,j}) \\
 & b_{i+1,j}? \arrow[ur, "1"] \arrow[dr, "0"'] & & b_{i+1,j+1}? \arrow[ur, "1"] \arrow[dr, "0"'] & \\
 & & (N_{i+1,j}) & & (N_{i+1,j+1})
\end{tikzcd}$$

Figure 6 Fragment for the Decision-DNNF in Proposition 5.6. The 0-gates are certified with clauses $C_{i,j}$ from Definition 5.5.

For $i < n$, we add $N_{i,j}$ according to Figure 6. The leaves of this proof fragment are either 0-gates that are certified by some clause $C^1_{i,j}$ to $C^4_{i,j}$, or are connected to some previously constructed gate $N_{i+1,j}$ or $N_{i+1,j+1}$, after making sure that the corresponding node contains no pebbles. In this way, we can obtain a graph that contains an appropriate gate $N_{i,j}$ for every node $P_{i,j}$.

Finally, we build the complete Decision-DNNF D. For each node $P_{i,j}$, ordered from least to greatest i, we decide $w_{i,j}$ and, if it is 0, also $b_{i,j}$. If both are 0, we connect to $N_{i,j}$; if both are 1 connect to a 0-gate certified by $C^9_{i,j}$. We merge the branches of all other cases and continue with the next node. After all nodes have been handled, we finally arrive at a single 1-gate.

Because of the ordering of the nodes, each path through D can decide each variable at most once. Therefore, D is indeed a Decision-DNNF. It is equivalent to φ and φ-certified. For each node $P_{i,j}$ we add at most 13 gates, and there is one additional 1-gate. In total, the number of gates is at most $13m + 1 = O(|\mathsf{PEB}_n|)$. ◀

The actual lower bound for MICE is the more challenging part.

▶ **Theorem 5.7.** *PEB_n requires* MICE *proofs of size* $2^{\Omega(n)}$.

Note that all known lower bounds for MICE so far are based on formulas with large Decision-DNNF representation [11]. However, this lower bound technique does not work for PEB_n as it has polynomial-sized Decision-DNNFs (Proposition 5.6).

Proof. Let π be a MICE proof of PEB_n. For a claim $I = (F, A)$, the set of *active* nodes $\mathcal{A}(I)$ contains exactly the nodes $P_{i,j}$ where $w_{i,j} \in \mathsf{vars}(F)$ or $b_{i,j} \in \mathsf{vars}(F)$. We define the *width* of I as $w(I) = |\mathcal{A}(F)|$ and the *boundary* of I as $B(I) := \{N \in \mathcal{A}_F \mid$ there is an adjacent node $N' \notin \mathcal{A}_F\}$.

We show that a claim that considers about half of all nodes also needs to have a large boundary:

▷ Claim 5.8. Let I be a claim with width bounded by $\frac{m}{3} \leq w(I) \leq \frac{2 \cdot m}{3}$. Then, $|B(I)| \geq \frac{n}{8}$.

Let $G_\pi = (V, E)$ be the representation of π as proof graph, i.e., V is the set of all claims in π and there is an edge (I_1, I_2) between two claim exactly if I_1 was used to derive I_2. Any claim $I \in V$ that is derived with (Join) has two incoming edges. For any such node, we delete the edge from the child that has the smaller width. We refer to the resulting graph as $G'_\pi = (V, E')$.

▷ Claim 5.9. For every model $\alpha \models \mathsf{PEB}_n$, there is a path π_α from the final claim to an (Ax) claim, along edges in E', such that for every every claim $(F, A) \in \pi_\alpha$, $\alpha \models A$.

In the rest of the proof, we use the π_α from Claim 5.9 and define V' to be the union of all π_α. We argue that claims in V' with a large boundary also have a large assumption:

▷ Claim 5.10. Any claim $I = (F, A) \in V'$ satisfies $|A| \geq |B(I)|$.

Next, we partition the claims of V' into two sets

$$X = \{I \in V' \mid w(I) < \frac{2}{3} \cdot m\},$$
$$Y = \{I \in V' \mid w(I) \geq \frac{2}{3} \cdot m\}.$$

Further, we define the set $S \subseteq Y$ as the set of nodes in Y that have a child in X, i.e.

$$S = \{I \in Y \mid \exists I_1 \in X : (I_1, I) \in E'\}.$$

We argue that all claims in S have large assumptions:

▷ Claim 5.11. Any claim $I = (F, A) \in S$ satisfies $|A| \geq \frac{n}{8}$.

On the other hand, every model of PEB_n corresponds to a claim in S:

▷ Claim 5.12. Let α be a model of PEB_n. Then, there is a claim $(F, A) \in S$ such that α and A are consistent.

Using Claims 5.11 and 5.12, we can finally prove the lower bound for the theorem. It is easy to observe that PEB_n has 2^m models because there are 2 satisfying assignments to the variables of each node. Let α be an assumption with at least $\frac{n}{8}$ variables and s the number of nodes with one or two variables in α. We observe that $2s \geq \frac{n}{8}$ and the number of models consistent with α is at most $1^s \cdot 2^{m-s} \leq 2^m \cdot 2^{n/16}$. Because each of 2^m models is consistent with a claim in S, S has at least $2^{n/16}$ elements. We conclude that $|\pi| \geq |V'| \geq |S| \geq 2^{n/16} = 2^{\Omega(n)}$ leading to the theorem. ◀

By combining Proposition 5.6 and Theorem 5.7, we finally obtain the separation of kcps from MICE:

▶ **Corollary 5.13.** kcps *is exponentially separated from* MICE.

While MICE and kcps are incomparable (Corollary 5.4, Corollary 5.13), kcps^+ simulates both systems (Theorem 4.1, Observation 4.2), which immediately leads to the following two separations.

▶ **Corollary 5.14.** kcps^+ *is exponentially separated from* MICE *and from* kcps.

With that, we have proven all separations from Figure 2.

6 Conclusion and future work

In this paper, we compare the strength of existing proof systems for #SAT. We mention that four of the systems we study, namely MICE, kcps^+, $\mathsf{CPOG}^{\mathsf{Decision\text{-}DNNF}}$ and CPOG, include propositional resolution derivations in proofs. These resolution derivations are needed to check propositional entailment steps. We could define variants of the four mentioned proof systems by replacing all resolution proofs by proofs in a different propositional proof system P (and in the extreme case even with NP oracle calls). Close inspection of our results shows that all simulations and separations as depicted in Figure 2 will continue to hold when resolution is replaced throughout by an arbitrary proof system P that is at least as strong as resolution (or an NP oracle).

We discuss a few directions for further work. From a *practical perspective*, our simulation results imply that CPOG might indeed be a suitable choice for proof logging as it simulates all other #SAT proof systems. But also $\mathsf{CPOG}^{\mathsf{Decision\text{-}DNNF}}$ or kcps^{+} could be practically sufficient for proof logging for all state-of-the-art #SAT solvers (and as of now, neither of these is known to be strictly weaker than CPOG).

In a related direction, we ask whether state-of-the-art knowledge compilers could effectively take advantage of kcps^{+} by using resolution instead of strictly relying on existing input clauses for certificates. We see this especially in the light that component caching-based #SAT solvers, which can already be captured with MICE, can be directly turned into practically effective knowledge compilers [38]. Hence, one might ask whether we can design even stronger knowledge compilers. Alternatively, we may use kcps or CPOG to certify caching-based #SAT solvers by emitting a Decision-DNNF.

From a *theoretical perspective* the system kcps^{+} appears quite interesting as it has an easy definition and is still strong enough to capture the different approaches of MICE and kcps. Designing a designated lower bound technique for kcps^{+} appears to be an interesting problem.

References

1 Handbook of satisfiability, 2021.

2 S. Akshay, Supratik Chakraborty, and Kuldeep S. Meel. Auditable Algorithms for Approximate Model Counting. In *AAAI'24*, 2024.

3 Michael Alekhnovich, Jan Johannsen, Toniann Pitassi, and Alasdair Urquhart. An exponential separation between regular and general resolution. *Theory Comput.*, 3(1):81–102, 2007. doi:10.4086/toc.2007.v003a005.

4 Albert Atserias, Johannes Klaus Fichte, and Marc Thurley. Clause-learning algorithms with many restarts and bounded-width resolution. *JAIR*, 40:353–373, 2011. doi:10.1613/jair.3152.

5 Fahiem Bacchus, Shannon Dalmao, and Toniann Pitassi. Algorithms and complexity results for #sat and bayesian inference. In *FOCS'03*, pages 340–351, 2003. doi:10.1109/SFCS.2003.1238208.

6 Teodora Baluta, Zheng Leong Chua, Kuldeep S. Meel, and Prateek Saxena. Scalable quantitative verification for deep neural networks. In *ICSE'21*, pages 312–323, 2021. doi:10.1109/ICSE43902.2021.00039.

7 Paul Beame, Henry A. Kautz, and Ashish Sabharwal. Towards understanding and harnessing the potential of clause learning. *JAIR*, 22:319–351, 2004. doi:10.1613/jair.1410.

8 Eli Ben-Sasson, Russell Impagliazzo, and Avi Wigderson. Near optimal separation of tree-like and general resolution. *Combinatorica*, 24(4):585–603, 2004. doi:10.1007/s00493-004-0036-5.

9 Olaf Beyersdorff. Proof complexity of quantified Boolean logic – a survey. In *Mathematics for Computation (M4C)*, pages 353–391. World Scientific Publishing Company, Singapore, 2022.

10 Olaf Beyersdorff and Benjamin Böhm. Understanding the relative strength of QBF CDCL solvers and QBF resolution. *LMCS*, 19(2), 2023. doi:10.46298/lmcs-19(2:2)2023.

11 Olaf Beyersdorff, Tim Hoffmann, and Luc Nicolas Spachmann. Proof complexity of propositional model counting. In *SAT'23*, volume 271, pages 2:1–2:18, 2023. doi:10.4230/LIPIcs.SAT.2023.2.

12 Olaf Beyersdorff, Tim Hoffmann, and Luc Nicolas Spachmann. Proof complexity of propositional model counting. *ECCC*, pages TR24–030, 2024. URL: https://eccc.weizmann.ac.il/report/2024/030.

13 Olaf Beyersdorff, Mikolás Janota, Florian Lonsing, and Martina Seidl. Quantified Boolean formulas. In *Handbook of Satisfiability*, pages 1177–1221. IOS Press, 2021. doi:10.3233/FAIA201015.

14 Maria Luisa Bonet, Juan Luis Esteban, Nicola Galesi, and Jan Johannsen. Exponential separations between restricted resolution and cutting planes proof systems. In *FOCS'98*, pages 638–647, 1998. doi:10.1109/SFCS.1998.743514.

15 Randal E. Bryant, Wojciech Nawrocki, Jeremy Avigad, and Marijn J. H. Heule. Certified knowledge compilation with application to verified model counting. In *SAT'23*, volume 271, pages 6:1–6:20, 2023. doi:10.4230/LIPIcs.SAT.2023.6.

16 Sam Buss and Jakob Nordström. Proof complexity and SAT solving. In *Handbook of Satisfiability*, pages 233–350. IOS Press, 2021. doi:10.3233/FAIA200990.

17 Florent Capelli. Understanding the complexity of #sat using knowledge compilation. In *LICS'17*, pages 1–10, 2017. doi:10.1109/LICS.2017.8005121.

18 Florent Capelli. Knowledge compilation languages as proof systems. In *SAT'19*, volume 11628, pages 90–99, 2019. doi:10.1007/978-3-030-24258-9_6.

19 Leroy Chew and Marijn J. H. Heule. Relating existing powerful proof systems for QBF. In *SAT'22*, volume 236, pages 10:1–10:22, 2022. doi:10.4230/LIPIcs.SAT.2022.10.

20 Stephen A. Cook. The Complexity of Theorem-Proving Procedures. In *Logic, Automata, and Computational Complexity: The Works of Stephen A. Cook*, volume 43, pages 143–152. ACM Books, 2023. doi:10.1145/3588287.3588297.

21 Stephen A. Cook and Robert A. Reckhow. The relative efficiency of propositional proof systems. *JSL*, 44(1):36–50, 1979. doi:10.2307/2273702.

22 Adnan Darwiche. Compiling knowledge into decomposable negation normal form. In *IJCAI'99*, pages 284–289, 1999. URL: http://ijcai.org/Proceedings/99-1/Papers/042.pdf.

23 Adnan Darwiche. On the tractable counting of theory models and its application to truth maintenance and belief revision. *JANCL*, 11(1-2):11–34, 2001. doi:10.3166/jancl.11.11-34.

24 Adnan Darwiche. A compiler for deterministic, decomposable negation normal form. In *AAAI'02*, pages 627–634, 2002. URL: http://www.aaai.org/Library/AAAI/2002/aaai02-094.php.

25 Adnan Darwiche and Pierre Marquis. A knowledge compilation map. *JAIR*, 17:229–264, 2002. doi:10.1613/jair.989.

26 Leonardo Dueñas-Osorio, Kuldeep S. Meel, Roger Paredes, and Moshe Y. Vardi. Counting-based reliability estimation for power-transmission grids. In *AAAI'17*, pages 4488–4494, 2017. doi:10.1609/aaai.v31i1.11178.

27 Johannes K. Fichte, Markus Hecher, and Florim Hamiti. The model counting competition 2020. *JEA*, 26(1):1–26, 2021. doi:10.1145/3459080.

28 Johannes Klaus Fichte, Markus Hecher, and Valentin Roland. Proofs for propositional model counting. In *SAT'22*, volume 236, pages 30:1–30:24, 2022. doi:10.4230/LIPIcs.SAT.2022.30.

29 Allen Van Gelder. Verifying RUP proofs of propositional unsatisfiability. In *ISAIM'08*, 2008.

30 Evguenii I. Goldberg and Yakov Novikov. Verification of proofs of unsatisfiability for CNF formulas. In *DATE*, pages 10886–10891, 2003. doi:10.1109/DATE.2003.10008.

31 Carla P. Gomes, Ashish Sabharwal, and Bart Selman. Model counting. In *Handbook of Satisfiability*, volume 336, pages 993–1014. IOS Press, 2021. doi:10.3233/FAIA201009.

32 Amin Haken. The intractability of resolution. *TCS*, 39:297–308, 1985. doi:10.1016/0304-3975(85)90144-6.

33 Markus Hecher, Patrick Thier, and Stefan Woltran. Taming High Treewidth with Abstraction, Nested Dynamic Programming, and Database Technology. In *SAT'20*, volume 12178, pages 343–360, 2020. doi:10.1007/978-3-030-51825-7_25.

34 Marijn Heule, Warren A. Hunt Jr., and Nathan Wetzler. Verifying refutations with extended resolution. In *CADE'13*, pages 345–359, 2013. doi:10.1007/978-3-642-38574-2_24.

35 Marijn J. H. Heule, Martina Seidl, and Armin Biere. Solution validation and extraction for QBF preprocessing. *JAR*, 58(1):97–125, 2017. doi:10.1007/s10817-016-9390-4.

36 Mikolás Janota and Joao Marques-Silva. Expansion-based QBF solving versus Q-resolution. *TCS*, 577:25–42, 2015. doi:10.1016/j.tcs.2015.01.048.

37 Stasys Jukna. *Boolean Function Complexity – Advances and Frontiers*, volume 27. Springer, 2012. doi:10.1007/978-3-642-24508-4.

38 Rafael Kiesel and Thomas Eiter. Knowledge Compilation and More with SharpSAT-TD. In *KR'23*, pages 406–416, 2023. doi:10.24963/kr.2023/40.

39 Benjamin Kiesl, Adrián Rebola-Pardo, and Marijn J. H. Heule. Extended resolution simulates DRAT. In *IJCAR'18)*, volume 10900, pages 516–531, jul, 2018. Held as Part of the Federated Logic Conference (FloC'18). doi:10.1007/978-3-319-94205-6_34.

40 Benjamin Kiesl, Adrián Rebola-Pardo, Marijn J. H. Heule, and Armin Biere. Simulating strong practical proof systems with extended resolution. *JAR*, 64(7):1247–1267, 2020. doi:10.1007/s10817-020-09554-z.

41 Hans Kleine Büning and Theodor Lettman. *Propositional logic: deduction and algorithms*. Cambridge University Press, New York, NY, USA, 1999.

42 Tuukka Korhonen and Matti Järvisalo. Integrating tree decompositions into decision heuristics of propositional model counters (short paper). In *CP'21*, pages 8:1–8:11, 2021. doi:10.4230/LIPIcs.CP.2021.8.

43 Jan Krajíček. *Proof complexity*, volume 170 of *Encyclopedia of Mathematics and Its Applications*. Cambridge University Press, 2019.

44 Jean-Marie Lagniez and Pierre Marquis. An improved decision-DDNF compiler. In *IJCAI'17*, pages 667–673, Melbourne, VIC, Australia, 2017. doi:10.24963/ijcai.2017/93.

45 Anna L. D. Latour, Behrouz Babaki, Anton Dries, Angelika Kimmig, Guy Van den Broeck, and Siegfried Nijssen. Combining stochastic constraint optimization and probabilistic programming - from knowledge compilation to constraint solving. In *CP'17*, volume 10416, pages 495–511, 2017. doi:10.1007/978-3-319-66158-2_32.

46 Christian J. Muise, Sheila A. McIlraith, J. Christopher Beck, and Eric I. Hsu. Dsharp: Fast d-DNNF Compilation with sharpSAT. In *AI'12*, volume 7310, pages 356–361. Springer Verlag, 2012. doi:10.1007/978-3-642-30353-1_36.

47 Tobias Philipp and Adrián Rebola-Pardo. DRAT proofs for XOR reasoning. In *JELIA'16*, pages 415–429, 2016.

48 Knot Pipatsrisawat and Adnan Darwiche. On the power of clause-learning SAT solvers as resolution engines. *Artificial Intelligence*, 175(2):512–525, 2011. doi:10.1016/j.artint.2010.10.002.

49 Dan Roth. On the hardness of approximate reasoning. *Artificial Intelligence*, 82(1-2):273–302, 1996. doi:10.1016/0004-3702(94)00092-1.

50 Nathan Segerlind. The complexity of propositional proofs. *Bulletin of Symbolic Logic*, 13(4):417–481, 2007. doi:10.2178/bsl/1203350879.

51 Weijia Shi, Andy Shih, Adnan Darwiche, and Arthur Choi. On tractable representations of binary neural networks. In *KR'20*, pages 882–892, 2020. doi:10.24963/kr.2020/91.

52 Marc Thurley. sharpSAT – Counting Models with Advanced Component Caching and Implicit BCP. In *SAT'06*, pages 424–429, 2006. doi:10.1007/11814948_38.

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

54 G. C. Tseitin. On the complexity of derivations in propositional calculus. In *Studies in Mathematics and Mathematical Logic, Part II*, pages 115–125. Springer, 1968.

55 Leslie G. Valiant. The complexity of computing the permanent. *TCS*, 8(2):189–201, 1979. doi:10.1016/0304-3975(79)90044-6.

56 Marc Vinyals. Hard examples for common variable decision heuristics. In *AAAI'20*, 2020.

57 Marc Vinyals, Jan Elffers, Jan Johannsen, and Jakob Nordström. Simplified and improved separations between regular and general resolution by lifting. In *SAT'20*, pages 182–200, 2020. doi:10.1007/978-3-030-51825-7_14.

58 Nathan Wetzler, Marijn Heule, and Warren A. Hunt Jr. Drat-trim: Efficient checking and trimming using expressive clausal proofs. In *SAT'14*, volume 8561, pages 422–429, 2014. doi:10.1007/978-3-319-09284-3_31.

59 Ennan Zhai, Ang Chen, Ruzica Piskac, Mahesh Balakrishnan, Bingchuan Tian, Bo Song, and Haoliang Zhang. Check before you change: Preventing correlated failures in service updates. In *NSDI'20*, pages 575–589, 2020. URL: https://www.usenix.org/conference/nsdi20/presentation/zhai.