Electronic Colloquium on Computational Complexity, Report No. 127 (2025)

Proof Systems That Tightly Characterise Model Counting Algorithms

Olaf Beyersdorff
Friedrich Schiller University Jena, Germany

Tim Hoffmann
Friedrich Schiller University Jena, Germany

Kaspar Kasche
Friedrich Schiller University Jena, Germany

Abstract

Several proof systems for model counting have been introduced in recent years, mainly in an attempt to model #SAT solving and to allow proof logging of solvers. We reexamine these different approaches and show that: (i) with slight adaptations, the conceptually quite different proof models of the dynamic system MICE [18] and the static system of annotated Decision-DNNFs [9] are equivalent and (ii) they tightly characterise state-of-the-art #SAT solving. Thus, these proof systems provide a precise and robust proof-theoretic underpinning of current model counting. We also propose new strengthenings of these proof systems that might lead to stronger model counters.

2012 ACM Subject Classification Theory of computation → Proof complexity

Keywords and phrases model counting, #SAT, proof complexity, proof systems, lower bounds, knowledge compilation

Funding *Olaf Beyersdorff*: Carl-Zeiss Foundation and DFG grant BE 4209/3-1
Tim Hoffmann: Carl-Zeiss Foundation
Kaspar Kasche: Carl-Zeiss Foundation

1 Introduction

Given a propositional formula φ, the model counting problem #SAT asks how many satisfying assignments exist for φ. This generalises the famous SAT problem, both in terms of computational complexity – where we see a jump from NP [12] to #P, encompassing almost all of the polynomial hierarchy [29] – and in terms of applications [1, 23, 27, 2, 32, 16]. For SAT solving, the main breakthrough happened in the late 1990s with the introduction of conflict-driven clause learning (CDCL) [24]. Since then, SAT solving has matured into a very successful and in terms of applications extremely versatile technology [5]. Model counting in comparison is in an earlier stage than SAT. There are two different problem settings: exact counting, considered here, and approximate counting [10]. These admit fundamentally different algorithmic approaches. Several state-of-the-art model counters, including D4 [22], DSHARP [25] and sharpSAT [28], successfully count on a large variety of formulas and regularly participate in the annual model counting competition [17].

One of the standard approaches as described by Capelli, Lagniez, and Marquis [9] is to use the classical DPLL branching algorithm for SAT and optimise it for counting via formula caching and decomposition into variable-disjoint subformulas.

From a theoretical point of view, the enormous success of SAT and #SAT solvers poses an intricate problem: why are these algorithms so successful and where are their limitations? Proof complexity provides one of the primary approaches towards this problem. For SAT, seminal work of Pipatsrisawat and Darwiche [26] has tightly characterised CDCL (with freely choosable heuristics) by propositional resolution. This implies that lower bounds for resolution proof size translate to lower bounds for the running time of CDCL. Here we pursue

ISSN 1433-8092

a similar characterisation for the classical DPLL-based #SAT approach that we mentioned above.

A second connection comes through proof logging whereby SAT runs for unsatisfiable formulas are efficiently mapped to resolution or in fact optimised, stronger proof systems [31]. Recently, this has also been studied for #SAT and a number of proof systems for proof logging have been proposed, including MICE [18], CPOG [6], and a number of systems using annotated Decision-DNNFs [8, 9].

For model counting, the connection to proof complexity was preceded by the discovery that #SAT solvers are intricately linked to knowledge compilation and in fact, Decision-DNNFs – the standard circuit format in knowledge compilation [19] – can be extracted from almost all model counters [7]. However, the DNNFs themselves are not sufficient for certification as it is hard to check whether a DNNF corresponds to the input CNF. This was realised by Capelli [8] who introduced the first #SAT proof system kcps using annotated Decision-DNNFs, and subsequent proof systems [9, 6] follow this approach. As these systems all use annotated circuits, they are static proof systems. The only exception is the line-based proof system MICE [18, 4], but this system also allows for efficient extraction of Decision-DNNFs [4]. The relative strength of these different calculi was recently determined by Beyersdorff et al. [3].

Our contributions. We summarise our main findings.

A. Characterising solver running time by proof size. Our first contribution is a tight characterisation of DPLL-style model counting in the framework of Capelli, Lagniez, and Marquis [9] (called KC^{Alg}_{Syn} here) by a new proof system that we call KC^{PS}_{Syn}. This builds on this prior work of Capelli et al. who constructed certifiable Decision-DNNFs from runs of the solvers. While it is not clear if these Decision-DNNFs characterise runtime of solvers, we show that this holds for our new proof system KC^{PS}_{Syn}, a slightly adapted version of certifiable Decision-DNNFs. This provides an analogue of the characterisation of CDCL by resolution [26].

B. Equivalence to augmented MICE. We further show that our new proof system KC^{PS}_{Syn} is quite robust and natural as it is equivalent to a version of MICE augmented by one natural rule for caching (or more formally, syntactic formula substitution). While the original system MICE is exponentially weaker than KC^{PS}_{Syn} (Theorem 4.3), MICE with this rule precisely captures KC^{PS}_{Syn} and therefore also the solving approach KC^{Alg}_{Syn} (Theorem 4.6). This is quite surprising as these systems use very different approaches.

C. Investigating stronger caching. Modern #SAT solvers apply formula caching in a syntactical setting. We investigate semantic caching – checking formulas for semantic equivalence – and show that the corresponding proof system KC^{PS}_{Sem} characterises solving with semantic caching (Theorem 5.3) and is also equivalent to MICE with a semantic substitution rule (Theorem 5.4). Further, we show that KC^{PS}_{Sem} proof size (neglecting the size of propositional annotations) of a CNF φ is characterised by the Decision-DNNF representation size of φ (cf. Proposition 5.7). This indicates that semantic caching is quite powerful and may be worth investigating in a practical setting.

Organisation. In Section 2 we review notions from proof complexity and knowledge representation. Sections 3, 4, and 5 contain our results described under A, B, and C above, respectively. We conclude in Sections 6 and 7 with an overview of the emerging landscape of #SAT proof systems (Figures 8 and 9) and outline questions for future research.

2 Preliminaries

We review notions from propositional logic and knowledge compilation. For $n \in \mathbb{N}$, let $[n] := \{1, 2, \ldots, n\}$.

In **propositional logic**, a variable v or its negation $\overline{v}$ is called a *literal* l. A *clause* is a disjunction of literals. A conjunction of clauses is a *conjunctive normal form* (CNF). Using Tseitin transformations [30], any formula can be efficiently translated into CNF.

The set of all variables in a formula F is denoted as $\mathsf{vars}(F)$. A (partial) *assignment* for a set of variables V is a (partial) function $\alpha : V \to \{0, 1\}$ mapping variables to Boolean values. The set of all $2^{|V|}$ complete assignments for V is denoted as $\langle V \rangle$. Given $V \subseteq \mathsf{vars}(F)$ and $\alpha \in \langle V \rangle$, $F[\alpha]$ represents the formula where all variables $x \in V$ are replaced by $\alpha(x)$. An assignment α *satisfies* F if $F[\alpha] = 1$, denoted $\alpha \models F$, and α *falsifies* F if $F[\alpha] = 0$. A formula F is *satisfiable* if there is some assignment $\alpha \in \langle \mathsf{vars}(F) \rangle$ with $\alpha \models F$. Otherwise, the formula is *unsatisfiable*.

Next, we introduce the *model counting problem* #SAT. Throughout the paper, we denote the CNF we want to count on as φ. Given φ, #SAT asks to compute $\#\varphi := |\mathsf{Mod}(\varphi)|$ with $\mathsf{Mod}(\varphi) := \{\alpha \in \langle \mathsf{vars}(\varphi) \rangle \mid \alpha \models \varphi\}$.

For a CNF F and set L of literals, we denote $cc(F, L)$ as the smallest *connected component*. Formally, it is the smallest set of clauses such that for every $C \in F$:

- if $\mathsf{vars}(C) \cap L \neq \emptyset$, then $C \in cc(F, L)$, and
- if there is some clause $D \in cc(F, L)$ with $\mathsf{vars}(C) \cap \mathsf{vars}(D) \neq \emptyset$, then $C \in cc(F, L)$.

We define *semantic consequence*. If for every assignment $\alpha \in \langle \mathsf{vars}(F) \rangle$ holds that $\alpha \models F$ implies $\alpha \models G$, we write $F \models G$. If $F \models G$ and $G \models F$, we write $F \equiv G$.

Any assignment can be seen as a CNF only consisting of unit clauses enforcing the assignment. Since a CNF is a set of clauses, set operations are well defined on CNFs and assignments. Two assignments are called *consistent* if they agree on their intersection.

A **proof system** for a language L is a polynomial time computable function f with range $\mathsf{rng}(f) = L$ [13]. In this paper L is always UNSAT or #SAT. We call π an f-proof for $x \in L$ if $f(\pi) = x$.

We use *simulations* to compare two proof systems P and Q for the same language. We say that P *p-simulates* Q if every Q-proof can be transformed in polynomial time into a P-proof of the same formula. If P and Q p-simulate each other, they are called *p-equivalent*, denoted $P \equiv Q$.

The best-studied proof system for UNSAT is *resolution*. It is a *line-based* proof system with clauses as proof lines. With the *resolution rule* we derive the clause $C \cup D$ from previous clauses $C \cup \{x\}$ and $D \cup \{\overline{x}\}$. A *resolution refutation* of a CNF is a derivation of the empty clause $\square$.

Knowledge compilation has been extensively studied [15, 20] and has a tight connections to model counting. Here we only need the notion of a Decision-DNNF.

A *circuit* C is a rooted directed acyclic graph with labelled nodes that we call *gates*. The set of all variables appearing in C is denoted as $\mathsf{vars}(C)$. For any gate $v \in C$, we denote the subcircuit with root v as C_v.

A Decision-DNNF, standing for *decision decomposable negation normal form* [14], is a circuit D that has one unique gate with in-degree 0 which is its root and represents the output of D. A simple Decision-DNNFs is illustrated in Figure 2. Any leaf of D, i.e. any gate with out-degree 0, is a 0-gate labelled with 0 or a 1-gate labelled with 1. Every gate in D that is not a leaf is either an AND-gate or a DECISION-gate.

An AND-gate v is labelled with an $\wedge$ and is *decomposable* which means that for any two child gates v_i and v_j of v, we have $\mathsf{vars}(D_{v_i}) \cap \mathsf{vars}(D_{v_j}) = \emptyset$. For simplicity, we assume w.l.o.g. that AND-gates have exactly two child nodes. A DECISION-gate is labelled with some variable x that is decided in this gate, i.e. there are two outgoing edges corresponding to the assignments $x = 0$ (dashed lines in our figures) and $x = 1$ (solid lines). Further, there must not be a path in D from the root to a leaf that decides the same variable more than once.

The main motivation for compiling CNFs to Decision-DNNFs is that on Decision-DNNFs various queries can be handled efficiently, in particular model counting and clause entailment:

▶ **Theorem 2.1** ([15]). *Let D be a Decision-DNNF and φ a CNF. Then:*

- *we can compute the model count of D in time $|D|^{O(1)}$,*
- *we can check $D \models \varphi$ in time $(|D| \cdot |\varphi|)^{O(1)}$.*

3 KC^{PS}_{Syn} proofs characterise #SAT solvers

3.1 Model counting algorithms

Many of the state-of-the-art knowledge compilers and model counters operate in a top-down fashion and can be seen as natural generalisations of SAT solvers based on DPLL. Following Capelli, Lagniez, and Marquis [9], we provide a high-level sketch of this solving approach which we call KC^{Alg}_{Syn}. The pseudocode in Algorithm 1 gives details.

The main idea of the solver is to build a decision tree that branches on variables until the remaining formula is satisfied or unsatisfiable. Soundness of a decision on x holds due to $F \equiv (x \wedge F[x]) \vee (\overline{x} \wedge F[\overline{x}])$ and we observe that $\#F = \#F[x] + \#F[\overline{x}]$. As the algorithm might visit the exact same formula multiple times we use a cache to keep track of formulas already computed. A further improvement can be obtained by checking if the formula consists of independent subformulas, i.e. if $F = F_1 \wedge F_2$ with $\mathsf{vars}(F_1) \cap \mathsf{vars}(F_2) = \emptyset$. In this case, we deal with the two formulas independently and have $\#F = \#F_1 \cdot \#F_2$. Finally, in order to check if the remaining formula is unsatisfiable, we use a SAT solver, typically based on CDCL. Clauses that are learned during a SAT call are saved as they might be helpful for subsequent SAT calls.

3.2 Certifiable Decision-DNNF circuits

The proof system *certifiable Decision-DNNF circuits* was proposed to certify KC^{Alg}_{Syn} [9]. For brevity, we refer to it as C-dec-DNNF. A C-dec-DNNF proof for a CNF φ is a restricted and annotated Decision-DNNF D such that we can verify $D \equiv \varphi$ efficiently. Conceptually, it is very similar to the proof system KC^{PS}_{Syn} which we define later in this section. In the following, we provide the formal definition of C-dec-DNNF.

For that, let φ be a CNF and D be a Decision-DNNF with root r. Additionally, we need a set R of clauses and a function e that assigns every gate $v \in D \setminus \{r\}$ one of its parent gates. We define $\mathsf{lits}(v)$ as the assignment of all literals decided on the unique path according to e from r to v. Further, let v, w be nodes in D such that there is an edge $e' = (w, v)$ and let F_w be some CNF that corresponds to w. We use the notation $F_w|_{e'}$ which we define as follows: If w is a DECISION-gate that leads to v by setting $x = 1$ (resp. 0), then $F_w|_{e'} := F_w[x]$ (resp. $F_w[\overline{x}]$). Otherwise, if w is an AND-gate, then $F_w|_{e'} := cc(F_w, \mathsf{vars}(D_v))$.

With that notation, we define F_v for every gate $v \in D$ inductively in a top-down fashion. For the root r, we define $F_r = \varphi$. Otherwise, for $w = e(v)$ with edge $e' = (w, v)$, we set $F_v = F_w|_{e'}$. With that, we can finally define the C-dec-DNNF proof system:

Algorithm 1 Pseudocode for KC^{Alg}_{Syn} [9]

procedure COMPILE(F, $L = \emptyset$, $R = \emptyset$)
  Data: **Input:** CNF F
    Input: set L of literals
    Input: set R of learnt clauses with $F \models R$
  Result: Decision-DNNF $D \equiv F[L]$
  if $F[L]$ has no clause **then return** new 1-gate
  if $F[L]$ has an empty clause **then return** new 0-gate
  if $R[L]$ has an empty clause **then return** new 0-gate
  Call a SAT solver on F with literals in L blocked
  Let R' be the clauses learnt by this call
  $R \leftarrow R \cup R'$
  if UNSAT(F) **then return** new 0-gate
  if cache($F[L]$) $\neq$ nil **then return** cache($F[L]$)
  $F' \leftarrow \{\, C \in F \mid C[L] \text{ is not satisfied} \,\}$
  if $F' = F_1 \wedge \cdots \wedge F_k$ with $k \geq 2$ and $\mathsf{vars}(F_i[L]) \cap \mathsf{vars}(F_j[L]) = \emptyset$ **then**
    $v \leftarrow$ new AND-gate
    for $i \leftarrow 1$ **to** k **do**
      $w_i \leftarrow$ COMPILE(F_i, L, R)
      connect v to w_i
  else
    Choose $x \in \mathsf{vars}(F[L])$
    $v \leftarrow$ new DECISION-gate deciding x
    for $\ell \in \{x, \neg x\}$ **do**
      $w \leftarrow$ COMPILE($F, L \cup \{\ell\}, R$)
      connect v to w with label ℓ
  cache($F[L]$) $\leftarrow v$
  return v

▶ **Definition 3.1** ([9]). *A C-dec-DNNF proof for a CNF φ is a 3-tuple (D, e, R) that satisfies the following conditions:*

(I) $D \models \varphi$,
(II) $R = \emptyset$ *or for every gate v that is not a 0-gate, F_v is satisfiable,*
(III) *for every gate $v \in D$ that is not a leaf and any edge $e' = (w, v)$, the formulas F_v and $F_w|_{e'}$ are syntactically equivalent,*
(IV) *R is a list of clauses that are each derived with a resolution step from φ or previous clauses in R,*
(V) *for every 0-gate $v \in D$ and any edge $e' = (w, v)$, either $F_w|_{e'}$ contains the empty clause, or w is a DECISION-gate and $R[\mathsf{lits}(w) \wedge l_{e'}]$ contains the empty clause where $l_{e'}$ is the literal that was decided for the edge e'.*

As already pointed out when C-dec-DNNF was introduced [9], condition (II) seems unnecessary, but it is required for the soundness of the system. We want to illustrate the underlying problem with the following small example: Let F be an unsatisfiable formula and $\varphi = (\overline{y} \wedge x \wedge F) \vee (x \wedge y)$ encoded as some CNF. Then, if we would not require (II), the Decision-DNNF in Figure 1 would be a valid C-dec-DNNF proof while $D \equiv \bot \not\equiv \varphi$.

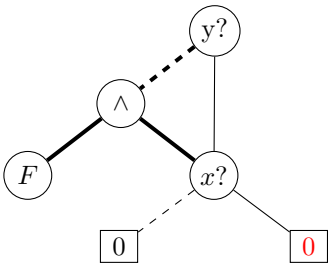

Figure 1 After a DECISION-gate, a dashed line corresponds to the assignment 0 and a solid line to 1. Thick lines are in the labelling e. We see that condition (II) is necessary: This Decision-DNNF D is unsatisfiable and thus not equivalent to $\varphi = (\overline{y} \wedge x \wedge F) \vee (x \wedge y)$ as the red 0 should be a 1. However, D satisfies all conditions except (II).

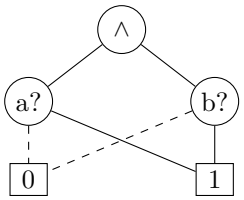

Figure 2 Algorithm KC^{Alg}_{Syn} is not able to find this Decision-DNNF D for $\varphi = a \wedge b \wedge (a \vee c) \wedge (a \vee \overline{c}) \wedge (b \vee c) \wedge (b \vee \overline{c})$ although $D \equiv \varphi$.

The main motivation for the introduction of C-dec-DNNF is its tight connection to KC^{Alg}_{Syn} which is captured in Theorem 3.2:

▶ **Theorem 3.2** ([9]). *Let R be a run of KC^{Alg}_{Syn} on a CNF φ. Then we can extract from R a C-dec-DNNF proof of φ in time $|R|$.*

This system was designed for certifying outputs of knowledge compilers. It is not obvious whether it characterises the runtime of KC^{Alg}_{Syn}, i.e. whether the converse of Theorem 3.2 holds. The question is: can KC^{Alg}_{Syn} compute a Decision-DNNF efficiently from a given C-dec-DNNF proof? For the corresponding simulation, we want to illustrate the main problem. Consider the formula $\varphi = a \wedge b \wedge (a \vee c) \wedge (a \vee \overline{c}) \wedge (b \vee c) \wedge (b \vee \overline{c})$. This formula is represented by the Decision-DNNF from Figure 2 and it is possible to construct a C-dec-DNNF proof based on this Decision-DNNF. However, the KC^{Alg}_{Syn} could never construct a circuit like this, as φ cannot be divided into independent subformulas without deciding c first. Thus, the C-dec-DNNF system seems to be slightly stronger than the KC^{Alg}_{Syn} algorithm.

3.3 The new proof system KC^{PS}_{Syn}

Now we introduce our new proof system KC^{PS}_{Syn} which can be seen as a restricted version of C-dec-DNNF such that the problem from Figure 2 disappears. In fact, we show that KC^{PS}_{Syn} characterises the runtime of KC^{Alg}_{Syn} (Theorem 3.7).

▶ **Definition 3.3** (KC^{PS}_{Syn}). *Let D be a Decision-DNNF, $\mathcal{F}$ be a labelling that assigns a CNF $\mathcal{F}(v)$ to every gate $v \in D$ and ρ be a set of resolution refutations. A KC^{PS}_{Syn} proof for a CNF φ is a 3-tuple $(D, \mathcal{F}, \rho)$ such that*

(i) $D \models \varphi$,

(ii) for any gate $v \in D$ that is not a 0-gate, $\mathcal{F}(v)$ is satisfiable,
(iii) $\mathcal{F}(r) = \varphi$ for the root $r \in D$,
(iv) for any AND-gate $v \in D$ with children s and t, $\mathcal{F}(s) = cc(\mathcal{F}(v), \mathsf{vars}(D_s))$ and analogously for $\mathcal{F}(t)$. Further, $\mathsf{vars}(\mathcal{F}(s)) \cap \mathsf{vars}(\mathcal{F}(t)) = \emptyset$,
(v) for any DECISION-gate $v \in D$ deciding variable x with children s if $x = 0$ and t if $x = 1$, $\mathcal{F}(s) = \mathcal{F}(v)[\overline{x}]$ and $\mathcal{F}(t) = \mathcal{F}(v)[x]$,
(vi) for any 0-gate $v \in D$ there is a resolution refutation of $\mathcal{F}(v)$ in ρ.

We want to point out that a KC^{PS}_{Syn} proof does not have to specify $\mathcal{F}$ explicitly as if it exists, it is uniquely determined by D and φ. Further, we can observe that KC^{PS}_{Syn} is very similar to C-dec-DNNF. The main difference is that instead of defining the labelling e, we label all gates directly with the corresponding CNF. As $\mathsf{lits}(v)$ is not defined any more, we cannot derive clauses in R; instead we include a resolution refutation for every 0-gate which proves that there are indeed no models we are missing. Potentially, we have to refute different 0-gates separately that could be handled with a single clause in R. However, the resulting overhead is at most polynomial.

For the proofs of this section, we need the invariant for KC^{PS}_{Syn} proofs from the following lemma:

▶ **Lemma 3.4.** *Let $(D, \mathcal{F}, \rho)$ be a KC^{PS}_{Syn} proof for a CNF φ. Further, let v be an arbitrary gate in D and P be any path from the root to v. Let β_v denote the assignment that corresponds to all decisions that are made on path P. Then, $\mathcal{F}(v) \subseteq \varphi[\beta_v]$.*

Note that the path P and thus the assignment β_v does not have to be unique. However, the claim holds for any β_v, no matter which path P we choose.

Proof. Let $v_1, v_2, \ldots, v_n$ with $v_1 = r$ and $v_n = v$ be the path P. We show the lemma by induction over n. The base case is obvious as $\beta_r = \emptyset$ and $\mathcal{F}(r) = \varphi$ as required by (iii). For the induction step, we assume that $\mathcal{F}(v_i) \subseteq \varphi[\beta_{v_i}]$. We distinguish what kind of gate v_i is.

Case 1. v_i is an AND-gate. Then, $\beta_{v_{i+1}} = \beta_{v_i}$ and $\mathcal{F}(v_{i+1}) = cc(\mathcal{F}(v_i), \mathsf{vars}(D_{v_{i+1}}))) \subseteq \mathcal{F}(v_i)$ by (iv). Thus, $\mathcal{F}(v_{i+1}) \subseteq \mathcal{F}(v_i) \subseteq \varphi[\beta_{v_i}] = \varphi[\beta_{v_{i+1}}]$.

Case 2. v_i is a DECISION-gate that w.l.o.g. sets $x = 1$ to get to v_{i+1}. Then, $\beta_{v_{i+1}} = \beta_{v_i} \wedge x$ and $\mathcal{F}(v_{i+1}) = \mathcal{F}(v_i)[x]$ by (v). Thus, $\mathcal{F}(v_{i+1}) = \mathcal{F}(v_i)[x] \subseteq \varphi[\beta_{v_i} \wedge x] = \varphi[\beta_{v_{i+1}}]$. ◀

Before we further analyse KC^{PS}_{Syn}, we show that is indeed a proof system in the sense of Cook and Reckhow [13]:

▶ **Proposition 3.5.** *KC^{PS}_{Syn} is a complete and sound proof system for #SAT.*

Proof. We start with the *soundness* of the system. Let a KC^{PS}_{Syn} proof $(D, \mathcal{F}, \rho)$ for a CNF φ be given. We have to show that $D \equiv \varphi$. As (i) ensures $D \models \varphi$, we only have to prove $\varphi \models D$ or equivalently $\neg D \models \neg\varphi$. Let α be an arbitrary assignment that falsifies D. Then, there has to be some path P from the root of D to some 0-gate v such that every decision on P is consistent with α. Because of (vi), there is a resolution refutation ρ of $\mathcal{F}(v)$. Let β_v be the assignment corresponding to path P. Using Lemma 3.4, we obtain $\mathcal{F}(v) \subseteq \varphi[\beta_v]$. Thus, ρ is also a refutation of $\varphi[\beta_v]$ implying that the assignment β_v makes φ unsatisfiable. By construction of β_v, we have $\beta_v \subseteq \alpha$, i.e. α falsifies φ. With that, the soundness of KC^{PS}_{Syn} is shown.

The *completeness* of the system is easy to observe as we can construct a valid KC^{PS}_{Syn} proof for any CNF φ by choosing as Decision-DNNF a complete binary decision tree that decides every variable in $\mathsf{vars}(\varphi)$ on every path.

Finally, in order to be a proof system in the sense of Cook-Reckhow, there has to be some way to *verify* a proof efficiently, i.e. in polynomial time. Note that we can compute the model count efficiently given a valid KC^{PS}_{Syn} proof as $D \equiv \varphi$ and we can count efficiently on Decision-DNNFs as stated in Theorem 2.1. Further, condition (i), $D \models \varphi$, can be checked efficiently as also shown in Theorem 2.1. All other requirements are straightforward to check in polynomial time. ◀

In fact, the proof of Proposition 3.5 neither uses condition (ii) nor $\mathsf{vars}(\mathcal{F}(s)) \cap \mathsf{vars}(\mathcal{F}(t)) = \emptyset$ in (iv). Thus, KC^{PS}_{Syn} would be a proof system even without these requirements. However, we stick to these additional requirements as the first guarantees the natural invariant $D_v \equiv \mathcal{F}(v)$ for any node $v \in D$ (Proposition 3.6). The second one is needed to show the equivalence to KC^{Alg}_{Syn} (Theorem 3.7).

▶ **Proposition 3.6.** *Let $(D, \mathcal{F}, \rho)$ be a KC^{PS}_{Syn} proof. Then, for every gate $v \in D$, the circuit D_v is equivalent to $\mathcal{F}(v)$.*

Proof. Let $v \in D$ be some fixed gate and $v_1, ..., v_n$ be a path in D from the root $v_1 = r$ to $v_n = v$. We show inductively, that the proposition holds for every v_i for $i \in [n]$. For the base case, we have $v_1 = r$ and $\mathcal{F}_r = \varphi \equiv D$ as required by (iii) and the soundness of KC^{PS}_{Syn} (Proposition 3.5). Next, let $i \in [n-1]$ be given. By the induction hypothesis, we have $D_{v_i} \equiv \mathcal{F}(v_i)$. We distinguish what kind of gate v_i is.

Case 1. v_i is a DECISION-gate leading to v_{i+1} by w.l.o.g. setting $x = 1$. Because of (v), we have $D_{v_{i+1}} = D_{v_i}[x] \equiv \mathcal{F}(v_i)[x] = \mathcal{F}(v_{i+1})$.

Case 2. v_i is an AND-gate with children v_{i+1} and v' and variable partition $\mathsf{vars}(\mathcal{F}(v_i)) = \mathsf{vars}(\mathcal{F}(v_{i+1})) \dot\cup V'$. Let $\alpha' \in \langle V' \rangle$ be a satisfying assignment of the formula $cc(\mathcal{F}(v), V')$. Note that α' has to exist as $\mathcal{F}(v_i)$ would be unsatisfiable otherwise, which would contradict (ii). Then, $D_{v_{i+1}} = D_{v_i}[\alpha] \equiv \mathcal{F}(v_i)[\alpha] = \mathcal{F}(v_{i+1})$. ◀

Next, we get to the result that provides the main motivation for the modifications of C-dec-DNNF. That is, we can generate solver runs from KC^{PS}_{Syn} proofs efficiently:

▶ **Theorem 3.7.** *For each CNF φ, the minimal runtime of KC^{Alg}_{Syn} on φ (with freely choosable heuristics) is polynomially equivalent to the minimal KC^{PS}_{Syn} proof size of φ.*

Before we can prove Theorem 3.7, we start with some propositions and lemmas.

▶ **Proposition 3.8.** *C-dec-DNNF p-simulates KC^{PS}_{Syn}. Further, this simulation is linear-time.*

Proof. Let $\pi = (D, \mathcal{F}, \rho)$ be a KC^{PS}_{Syn} proof for a given CNF φ. Further, let e be an arbitrary labelling according to a C-dec-DNNF proof, i.e. e assigns every gate $v \in D$ a unique parent gate $e(v)$. We first show the following invariant:

Claim 1: For every gate $v \in D$ the corresponding formulas $\mathcal{F}(v)$ in KC^{PS}_{Syn} and F_v in C-dec-DNNF are the same.

That is, we show $F_v = \mathcal{F}(v)$ for any node $v \in D$ by induction in a top-down approach. For that, $r = v_1, v_2, \ldots, v_{n-1}, v_n = v$ be the unique path from the root r to v according to the labelling e. In the base case, $F_r = \varphi = \mathcal{F}(r)$ by using (iii) and the construction of F. Thus, we can assume that $F_{v_i} = \mathcal{F}(v_i)$ for some $i \in [n]$ and we have to show that $F_{v_{i+1}} = \mathcal{F}(v_{i+1})$. For that we distinguish what kind of gate v_i is.

Case 1. v_i is an AND-gate. Then, $F_{v_{i+1}} = cc(F_{v_i}, \mathsf{vars}(D_{v_{i+1}})) = cc(\mathcal{F}(v_i), \mathsf{vars}(D_{v_{i+1}})) = \mathcal{F}(v_{i+1})$ where we used the construction of F in C-dec-DNNF, the induction hypothesis and (iv).

Case 2. v_i is a DECISION-gate that leads to v_{i+1} by setting w.l.o.g. $x = 1$. Then, $F_{v_{i+1}} = F_{v_i}[x] = \mathcal{F}(v_i)[x] = \mathcal{F}(v_{i+1})$ where we used the construction of F in C-dec-DNNF, the induction hypothesis and (v).

Next, we show that D and e satisfy the requirements (I), (II) and (III) for C-dec-DNNF proofs:

- (I), $D \models \varphi$, is ensured by (i).
- (II), $\mathcal{F}(v)$ is satisfiable for any non-0-gate v, is guaranteed by (ii).
- For (III), let $v, w \in D$ be any gates such that there is an edge $e' = (w, v)$. We have to show that the formulas F_v and $F_w|_{e'}$ are syntactically equivalent. Because of Claim 1 from above, we have $F_v = \mathcal{F}(v)$ and $F_w = \mathcal{F}(w)$. We can use the exact same argument from the proof of Claim 1, i.e. if w is an AND-gate, we use (iv) and if it is a DECISION-gate, we use (v).

Therefore, we only have to describe how we choose R in order to ensure that (IV) and (V) are satisfied. For that, we do the following for any 0-gate v: Let β_v be the assignment of all literals decided on the path defined by e from v to the root of D. Because of (v), v is labelled with some refutation ρ_v showing $\mathcal{F}(v) \vdash \square$. With Lemma 3.4, we have $\mathcal{F}(v) \subseteq \varphi[\beta_v]$, i.e. ρ_v can also be used as derivation for $\varphi[\beta_v] \vdash \square$. By weakening, we obtain a derivation $\varphi \vdash \overline{\beta}_v$. For any 0-gate v, we add the corresponding clause $\overline{\beta}_v$ and its derivation to R. Like that, we derive a clause for every 0-gate in order to satisfy (V). Further, we can add all required clauses for the derivations to R and satisfy (IV). Note that for any 0-gate v, we add $|\rho_v|$ clauses to R, i.e. the resulting proof size does not increase.

As all requirements for C-dec-DNNF are satisfied, $\pi' = (D, e, R)$ is indeed a valid C-dec-DNNF proof for φ. By construction, we have $|\pi'| = |\pi|$. ◀

As already pointed out in Figure 2, the algorithm KC^{Alg}_{Syn} cannot construct arbitrary Decision-DNNFs but only a restricted class of these circuits. In the following, we see that the constructed Decision-DNNFs are *formula-decomposable*. And in fact, if we add this restriction to C-dec-DNNF, the system becomes exactly as strong as KC^{PS}_{Syn}.

▶ **Definition 3.9.** *Let π be a C-dec-DNNF proof. We say that π is* formula-decomposable *if every AND-gate v satisfies: for its child nodes s and t we have* $\mathsf{vars}(F_s) \cap \mathsf{vars}(F_t) = \emptyset$.

Next, we show that formula-decomposable C-dec-DNNF is p-equivalent to KC^{PS}_{Syn}. For that, we show the two simulations separately.

▶ **Lemma 3.10.** *Formula-decomposable C-dec-DNNF p-simulates* KC^{PS}_{Syn}.

Proof. We use the exact same construction as in the proof of Proposition 3.8. It is sufficient to argue that the extracted C-dec-DNNF proof is formula-decomposable.

For that, we have a closer look at the proof of Proposition 3.8 and consider any AND-gate v with children s and t. With the second requirement in (iv), we have $\mathsf{vars}(\mathcal{F}(s)) \cap \mathsf{vars}(\mathcal{F}(t)) = \emptyset$. The invariant we showed in Claim 1, we obtain $F_s = \mathcal{F}(s)$ and $F_s = \mathcal{F}(s)$. Putting these two things together, we get $\mathsf{vars}(F_s) \cap \mathsf{vars}(F_t) = \emptyset$. Thus, the extracted proof is indeed formula-decomposable. ◀

Next, we show a helpful invariant for C-dec-DNNF proofs:

▶ **Lemma 3.11.** *Let $\pi = (D, e, R)$ be a formula-decomposable C-dec-DNNF proof for CNF φ. Further, let $v \in D$ be given. If there is a path $r = v_1, v_2, \ldots, v_n = v$ from the root r of D to v such that F_{v_i} is satisfiable for all $i \in [n]$, then $F_v \equiv D_v$.*

Proof. We show this by induction in a top-down fashion. In the base case, we have $F_r = \varphi \equiv D_r$ by the soundness of the C-dec-DNNF proof system. Thus, we assume that $v_{n-1} \equiv D_{v-1}$. We distinguish what kind of gate v_{i-1} is.

Case 1. v_{i-1} is a DECISION-gate leading to v_i by w.l.o.g. setting $x = 1$. Then, $F_{v_i} = F_{v_{i-1}}[x] \equiv D_{v_{i-1}}[x] = D_{v_i}$.

Case 2. v_{i-1} is an AND-gate with children v_i and v'. Then, we have $F_{v_{i-1}} = F_{v_i} \wedge F_{v'}$. As π is formula-decomposable, the variables of these formulas are disjoint, i.e. there is a partition of the variables $V^F = \mathsf{vars}(F_{v_{i-1}})$ with $V^F = V^F_{v_i} \dot\cup V^F_{v'}$ such that $\mathsf{vars}(F_{v_i}) \subseteq V^F_{v_i}$ and $\mathsf{vars}(F_{v'}) \subseteq V^F_{v'}$. Further, the AND-gate v_{i-1} is decomposable, i.e. there is a partition of the variables $V^D = \mathsf{vars}(D_{v_{i-1}})$ with $V^D = V^D_{v_i} \dot\cup V^D_{v'}$ such that $\mathsf{vars}(D_{v_i}) \subseteq V^D_{v_i}$ and $\mathsf{vars}(D_{v'}) \subseteq V^D_{v'}$. Next we show, that we can partition $V^F \cup V^D$ as follows:

$$V^F \cup V^D = (V^F_{v_i} \cup V^D_{v_i}) \dot\cup (V^F_{v'} \cup V^D_{v'}).$$

$V^F_{v_i}$ and $V^F_{v'}$ are disjoint by construction. Thus, we only have to show that $V^F_{v_i}$ and $V^D_{v'}$ do not share variables (the other cases are symmetric). For that, we assume that there is some variable x that appears in both $V^F_{v_i}$ and $V^D_{v'}$. Then, x appears in F_{v_i} and also in $F_{v_{i-1}}$ as $F_{v_{i-1}} = F_{v_i} \wedge F_{v'}$. By definition of C-dec-DNNF condition (III), we have $F_{v'} = cc(F_{v_{i-1}}, \mathsf{vars}(D_{v'}))$. x appears in both $F_{v_{i-1}}$ and $D_{v'}$, and thus also in $F_{v'}$. With that, we obtain $x \in V^F_{v'}$ which is a contradiction to $x \in V^F_{v_i}$. This shows that the partition above is indeed valid.

Since $F_{v_{i-1}}$ is satisfiable and $F_{v_{i-1}} \equiv D_{v_{i-1}}$, there is an assignment $\alpha \in \langle \mathsf{vars}(V^F) \cup \mathsf{vars}(V^D) \rangle$ that satisfies both $F_{v_{i-1}}$ and $D_{v_{i-1}}$. We restrict this assignment to $\beta := \alpha|_{V^F_{v'} \cup V^D_{v'}}$. Then,

$$F_{v_i} = F_{v_i}[\beta] \wedge F_{v'}[\beta] = F_{v_{i-1}}[\beta] \equiv D_{v_{i-1}}[\beta] = D_{v_i}[\beta] \wedge D_{v'}[\beta] = D_{v_i},$$

leading to the lemma. ◀

With this invariant, we can show the simulation in the other direction:

▶ **Lemma 3.12.** KC^{PS}_{Syn} *p-simulates formula-decomposable* C-dec-DNNF.

Proof. Let $\pi = (D, e, R)$ be a formula-decomposable C-dec-DNNF proof for CNF φ and let D' be the Decision-DNNF where we introduce additional 0-gates and 1-gates such that every leaf of D has a unique parent gate. Further, we extend e to some e' that also contains the edges from any leaf of D' to its unique parent. We define F_v and $\mathsf{lits}(v)$ for any leaf v according to e'. Note that like that (III) does also hold for edges leading to leaves. It is easy to observe that for any 0-gate v in D' holds that either F_v contains the empty clause or $R[\mathsf{lits}(v)]$ contains the empty clause.

D' may contain a gate v which is not a leaf such that F_v is unsatisfiable. Let v' be the first gate on the path from r to v that has a formula $F_{v'}$ that is unsatisfiable. We replace v' by a 0-gate. Let D'' be the resulting Decision-DNNF which satisfies that for every inner gate v, F_v is satisfiable. Note that $D \equiv D' \equiv D''$. The first equivalence is obvious, as copying some 0-gates or 1-gates does not change the semantics of the circuit. The second equivalence is due to Lemma 3.11.

For the KC^{PS}_{Syn} proof we choose the Decision-DNNF D'' and use $\mathcal{F}(v) = F_v$ as labels for the nodes. Then, D'' and $\mathcal{F}$ meet the conditions (i) - (v) for a KC^{PS}_{Syn} proof:

- (i), $D'' \models \varphi$, is satisfied: (I) ensures that $D \models \varphi$ and we have $D \equiv D''$ by construction.
- (ii), for any inner gate v, $\mathcal{F}(v)$ is satisfiable, is satisfied by the construction of D''.

- (iii), $\mathcal{F}(r) = \varphi$ is satisfied: By construction, we set $\mathcal{F}(r) = F_r = \varphi$ where r is the root of D''.
- (iv), the requirements for AND-gates, are satisfied: Let v'' be an AND-gate in D'' and s'' be a child of v''. These gates corresponds to some gates v and s in D. Then, $\mathcal{F}(s) = F_s = cc(F(v), \mathsf{vars}(D_s)) = cc(\mathcal{F}(v), \mathsf{vars}(D_s))$ where we used $\mathcal{F}(s) = F_s$, $\mathcal{F}(v) = F_v$ by construction and (III).
  Further, we have to show that $\mathsf{vars}(\mathcal{F}(s)) \cap \mathsf{vars}(\mathcal{F}(t)) = \emptyset$ where t is the second child of v. For that, we need that the C-dec-DNNF proof π is formula-decomposable: With that, we have $\emptyset = \mathsf{vars}(F_s) \cap \mathsf{vars}(F_t) = \mathsf{vars}(\mathcal{F}(s)) \cap \mathsf{vars}(\mathcal{F}(t))$.
  Thus, these properties do also hold for D''.
- (v), the requirements for DECISION-gates, are satisfied: Let v'' be a DECISION-gate leading to s'' in D'' by w.l.o.g. setting $x = 1$. Let v, s be the corresponding gates in D. Then, $\mathcal{F}(s) = F_s = F_v[x] = \mathcal{F}(v)[x]$. where we again used $\mathcal{F}(s) = F_s$, $\mathcal{F}(v) = F_v$ by construction and (III). Thus, this property does also hold for D''.

Therefore, we only have to construct the resolution refutations for the 0-gates to ensure (vi). Let v be any 0-gate in D'' and let w be its unique parent with edge $e = (w, v)$. There are three cases.

Case 1. If $\mathcal{F}(v) = F_v = F_w|_e$ contains the empty clause, there is nothing to do.

Case 2. Otherwise, it could be the case, that v is not a 0-gate in D' and we replaced it because F_v is unsatisfiable. Then, because of (II), we have $R = \emptyset$. Further, because of Lemma 3.11, we have $F_v \equiv D'_v$, i.e. D'_v has to be unsatisfiable as well. Thus, we can remove one child for any AND-gate in D'_v such that every leaf in D'_v is a 0-gate. The remaining circuit D'_v has only DECISION-gates and 0-gates. Since $R = \emptyset$, for every 0-gate v, F_v has to contain the empty clause. Therefore, D'_v is a binary decision tree that refutes F_v. It is well-known, that then also a resolution refutation of $F_v = \mathcal{F}(v)$ has to exist of size $|(D'_v)|$.

Case 3. Otherwise, $R[\mathsf{lits}(w) \wedge l_e] = R[\mathsf{lits}(v)]$ has to contain the empty clause because of (V). Let C be a clause in R that is falsified by the assignment $\alpha = \mathsf{lits}(v)$, i.e. in R we derive $\varphi \vdash C$ with resolution and $C[\alpha] = \square$. If we restrict every clause in this derivation to α, we have a derivation ρ_v for $\varphi[\alpha] \vdash \square$. Because of the construction (condition (iv)) of $\mathcal{F}$, we have $\varphi[\alpha] = \mathcal{F}_v \cup X$ where X is some (potentially empty) set of clauses that is not contained in $\mathcal{F}_v$ any more because of previous AND-gates. Thus, $\mathsf{vars}(\mathcal{F}_v) \cap \mathsf{vars}(X) = \emptyset$. If X is empty, then ρ_v is the derivation $F_v \vdash \square$ that we need for (v). Otherwise, v cannot be the root of D'' and on the path from r to v according to e, there has to be at least one AND-gate. Let $w_1, w_2, \ldots, w_{k-1}$ be the set of AND-gates on this path. These AND-gates partition the circuit D'' in subcircuits $D_{v_1}, D_{v_2}, \ldots, D_{v_k}$ such that $\mathsf{vars}(D_{v_i})$ are pairwise disjoint because of decomposability. W.l.o.g. let $\mathsf{vars}(F_v) \subseteq \mathsf{vars}(D_1)$ and $\mathsf{vars}(X) \subseteq \mathsf{vars}(D_{v_2}) \cup \cdots \cup \mathsf{vars}(D_{v_k})$. Since $R \neq \emptyset$, (II) ensures that all inner gates are satisfiable, v_i are all satisfiable and in particular $D_{v_i} \equiv F_{v_i}$ because of Lemma 3.11. Therefore, we obtain $F_v \subseteq F_{v_1}$ and $X = F_{v_2} \cup \cdots \cup F_{v_k}$. As F_{v_i} are satisfiable, let $\alpha_i \in \langle \mathsf{vars}(F_{v_i}) \rangle$ be a satisfying assignment of F_{v_i}. Then, $\alpha_X := \alpha_2 \wedge \cdots \wedge \alpha_k$ is a satisfying assignment of X with $\mathsf{vars}(\alpha_X) \cap \mathsf{vars}(F_v) = \emptyset$. Putting these things together, $\rho_v[\alpha_X]$ is a refutation of $\varphi[\alpha \wedge \alpha_X] = F_v = \mathcal{F}(v)$.

Let ρ be the set of ρ_v for all 0-gates v in D from cases 2 and 3. Since (v) is satisfied as well, $\pi' = (D'', \mathcal{F}, \rho)$ is indeed a valid KC^{PS}_{Syn} proof. By construction, $|D''| \leq 2 \cdot |D|$ and $|\rho_v| \leq |D'|$ (in case 2) or $|\rho_v| \leq |R|$ (in case 3) for any 0-gate v. It may happen that we have to derive very similar derivations ρ_v over and over again for different 0-gates v which are handled with a single derivation in R. Thus, the proof size in this simulation is not linear any more. We can estimate the proof size with $|\pi'| \leq |\pi|^2$. ◀

▶ **Lemma 3.13.** *Let R be a run of KC^{Alg}_{Syn} on some CNF φ. Then, the extracted C-dec-DNNF proof is w.l.o.g. formula-decomposable.*

Proof. We use the exact same C-dec-DNNF proof that is extracted in Capelli, Lagniez, and Marquis [9] and show that this proof is formula-decomposable.

This proof extraction is straightforward, as we use the Decision-DNNF D that is constructed by the algorithm. The labelling e corresponds to the path the algorithm takes, i.e. an edge (w, v) is in e, exactly if the algorithm comes from gate w and occurs at gate v for the first time. Also the set R of learnt clauses is in the proof the exact same from the algorithm.

To show that this extracted proof is formula-decomposable, we have a look at line 12 of KC^{Alg}_{Syn}. We can observe that when we create an AND-gate v, the formula F_v has to syntactically decompose in formulas with disjoint variables. ◀

With that, we can finally put everything together and prove Theorem 3.7:

Proof of Theorem 3.7. We start by proving that we can extract KC^{PS}_{Syn} proofs from a solver run efficiently. For that, we only have to put the results from above together: We first observe a C-dec-DNNF proof π according to Theorem 3.2. As shown in Lemma 3.13, π is formula-decomposable. Further, with Lemma 3.12, we can can construct a KC^{PS}_{Syn} proof from π efficiently.

For the other direction, i.e. that we can generate solver runs from proofs efficiently, let a KC^{PS}_{Syn} proof π for a CNF φ be given. The solver goes through the proof from the top, with the invariant that at every gate v, $F[L]$ in the solver is $\mathcal{F}(v)$ in the proof. On DECISION-gates, it branches on the corresponding variable, and on AND-gates it decomposes the formula. If it hits an already visited gate, it uses a syntactic cache lookup. When we reach a 1-gate, the corresponding formula has to be tautological, i.e. the empty CNF.

When we reach a 0-gate v, it is labelled with some resolution refutation ρ_v of the corresponding formula $\mathcal{F}(v) = F_v$. Let $\mathsf{lits}(v)$ be the term containing all literals corresponding to the decisions that were made in order to get to v. By construction, we have $\varphi[\mathsf{lits}(v)] = F_v \cup X$ for some formula X which may arises because of AND-gates. Therefore, ρ_v is also a derivation $\varphi[\mathsf{lits}(v)] \vdash \square$. By weakening all clauses of ρ_v with $\overline{\mathsf{lits}(v)}$, we obtain a derivation $\varphi \vdash \overline{\mathsf{lits}(v)} =: C$, i.e. the solver can learn this clause C and put it into the set R. It can reproduce the 0-gate in line 9. ◀

A further insight from the proof is that saving learnt clauses in R can give at most polynomial speed-up. Therefore, from a purely theoretical point of view and ignoring polynomial overhead, clause saving does not help. For practical purposes, of course, it might still help to improve performance.

4 KC^{PS}_{Syn} vs MICE

Next, we compare KC^{PS}_{Syn} to the existing MICE proof system [18, 4]. We briefly explain MICE and show that it is weaker than KC^{PS}_{Syn}. However, we can strengthen MICE by adding a natural derivation rule. With that rule, the adapted MICE becomes as strong as KC^{PS}_{Syn}.

4.1 The MICE proof system

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. Semantically, in our definition a claim (F, A, c) says that $F[A]$ has $c \cdot 2^{|\mathsf{vars}(F[A])|}$ models. 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 3 such that the final claim has the form $(\varphi, \emptyset, c)$ for some count c. Thus, for the final claim $\#\varphi = c \cdot 2^{|\mathsf{vars}(F)|}$. As the count c in a correct claim (F, A, c) is determined by F and A, we sometimes omit it and write (F, A) instead.

Axiom.

$$\frac{}{(\emptyset, \emptyset, 1)} \quad (\mathsf{Ax})$$

0-Axiom.

$$\frac{}{(F, A, 0)} \quad (\mathsf{0\text{-}Ax})$$

- (A-1) there is a resolution refutation of $F[A]$

2-Composition.

$$\frac{(F, A \wedge \{x = 0\}, c_1) \quad (F, A \wedge \{x = 1\}, c_2)}{(F, A, c_1 + c_2)} \quad (\mathsf{2\text{-}Comp})$$

- (C-1) $x \notin \mathsf{vars}(A)$

Join.

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

- (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.

$$\frac{(F_1, A_1, c_1)}{(F, A, c_1)} \quad (\mathsf{Ext})$$

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

Figure 3 Adapted inference rules for MICE, following Beyersdorff, Hoffmann, and Spachmann [4]. Bullet points indicate the conditions under which the rule is applicable.

This is a slight change compared to the prior definition of MICE, but does not materially impact the proof system. The rules become slightly easier as we get rid of the powers of 2 if the number of variables in the formula changes. Another small modification in our definition of MICE concerns the rules (0-Ax) and (2-Comp) which are special cases of a more general *composition* derivation rule in Beyersdorff, Hoffmann, and Spachmann [4]. As we use this slightly adapted version of MICE, we show in the following, that these adaptions do not change the strength of MICE.

In our version, a MICE claim (F, A, c) states that $F[A]$ has exactly $c \cdot 2^{|\mathsf{vars}(F[A])|}$ models, while in the MICE version from Beyersdorff, Hoffmann, and Spachmann [4], the claim (F, A, c) states that $F[A]$ has exactly c models. Therefore, in the original (Ext) rule, we can derive

$(F, A, c_1 \cdot 2^{|\mathsf{vars}(F)\setminus(\mathsf{vars}(F_1)\cup\mathsf{vars}(A))|})$ from (F_1, A_1, c_1) while in our version, we get rid of the power of two. It is easy to observe that this modification does not change the strength of the system.

Further, the original MICE version allows *Composition* in a more general setting with several claims while we only have the two restricted rules (0-Ax) and (2-Comp) instead. Proposition 4.1 shows that also this second modification doesn't change the system materially.

Composition.

$$\frac{(F, A_1, c_1) \quad \cdots \quad (F, A_n, c_n)}{(F, A, \sum_{i\in[n]} c_i)} \qquad (\mathsf{Comp})$$

- (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*.

Figure 4 The original (Comp) inference rule for MICE from Beyersdorff, Hoffmann, and Spachmann [4].

▶ **Proposition 4.1.** *The proof system* MICE *where the derivation rule (Comp) is replaced by (2-Comp) and (0-Ax) is p-equivalent to* MICE.

Note, that a very similar result was already used to show that we can extract a Decision-DNNF efficiently from a MICE proof [4].

Proof. The fact that the original MICE p-simulates the modified MICE is obvious, as both (2-Comp) and (0-Ax) are special cases of (Comp).

For the other direction it is sufficient to argue how to simulate the (Comp) rule. For that, we assume that in a MICE proof π some claim (F, A) is derived with (Comp) using previous claims $(F, A_1), (F, A_2), \ldots, (F, A_n)$ and the absence of models statement ρ.

If $n = 0$, i.e. the (Comp) application does not use any other claims, we can derive (F, A) with (0-Ax). For the required resolution refutation (A-1) we use ρ.

Otherwise let $n \geq 1$. Remember that (C-1) ensures $\mathsf{vars}(A_1) = \mathsf{vars}(A_2) = \cdots = \mathsf{vars}(A_n)$. Let $V = \mathsf{vars}(A) \setminus \mathsf{vars}(A_1)$ and let T be a complete binary decision tree that decides all variables in V, We associate every node v of T with claim $(F, A \wedge \alpha_v)$ where α_v is the assignment corresponding to the path from the root of T to v.

With that, any leaf v of T corresponds to an assignment $\alpha \in \langle V \rangle$. Every claim from $I_i = (F, A_i)$ for $i \in [n]$ that was used for the (Comp) step, corresponds to a claim $(F, A \wedge \alpha_v)$ for a unique leaf v_i. We replace T by the its minimised version that contains only the nodes v_i and the decision nodes in order to get there.

We can estimate the size of the resulting tree T as follows: It has size at most depth $|V| \leq \mathsf{vars}(\varphi)$. Further, it has n non-zero leaves and as we removed the unnecessary 0-gates, the resulting tree has at most $n \cdot \mathsf{vars}(\varphi)$ nodes.

In order to simulate the (Comp) step, we use use exactly the claims from T, listed in a bottom-up ordering. Next, we argue how to derive these claims from T:

- For the claims corresponding to the leaves v_i, we have already derived $I_i = (F, A_i)$ as we wanted to use it for the (Comp) step.

- For the claims corresponding to an other leaf v, the corresponding claim has to have the form $I_v = (F, A \wedge \alpha_v, 0)$. The count has to be zero, as otherwise, there would exist an assignment satisfying F, A and none of the A_i. But then the absence of models statement ρ, used for the (Comp) step, which is a refutation of $A \cup F \cup \{\overline{A}_i \mid i \in [n]\}$, could not exist. We can derive I_v with (0-Ax) by using ρ as proof that the count 0 is indeed correct.
- For any other claim, it corresponds to an inner gate v that decides some variable x with children $I_{\overline{x}}$ and I_x that are derived already. Then, we derive the corresponding claim with (2-Comp) using $I_{\overline{x}}$ and I_x.

Therefore, we can simulate the (Comp) rule by using at most $n \cdot \mathsf{vars}(\varphi)$ additional MICE claims. Further, we may use additional resolution refutations for the applications of (0-Ax). These cannot be larger than the refutation ρ used for the (Comp) step. Thus, the simulation of one (Comp) application requires at most $n \cdot |\mathsf{vars}(\varphi)| \cdot |\rho|$ steps. As the π can apply (Comp) a most n times, the simulation of all (Comp) steps increases the size of the proof at most by $n^2 \cdot |\mathsf{vars}(\varphi)| \cdot |\rho|$.

◀

4.2 A weakness of MICE

It is already known that there exist formulas with polynomial-size Decision-DNNFs while any MICE proof needs exponentially many steps [3]. However, it has remained open so far whether MICE allows for efficient proof logging for current solving techniques. It is conceivable that #SAT solvers produce Decision-DNNFs that are restricted enough to extract MICE proofs efficiently. We show that this is not the case, i.e. there exist formulas without short MICE proofs that state-of-the-art solvers can handle efficiently. This follows from our results that KC^{PS}_{Syn} characterises KC^{Alg}_{Syn} solving (Theorem 3.7), while KC^{PS}_{Syn} is stronger than MICE (Theorem 4.3). For the separation we use a variant of the pebbling formulas, well known in proof complexity:

▶ **Definition 4.2** ([3]). *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 $j \leq i$. PEB_n is a CNF defined as follows:*

- *For every $i, j \in [n-1], j \leq i$ 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

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

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

Next, we provide some intuition for these PEB_n formulas. They have nodes $P_{i,j}$ forming a pyramidal graph. For each node $P_{i,j}$ there are two variables $w_{i,j}$ and $b_{i,j}$. $w_{i,j}$ 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. 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.

Next, we show that these PEB_n formulas separate KC^{PS}_{Syn} and MICE:

▶ **Theorem 4.3.** *The PEB_n formulas have KC^{PS}_{Syn} proofs of size $O(n)$ while any MICE proof has size at least $2^{\Omega(n)}$.*

Before proving this theorem, we show the upper bound for KC^{PS}_{Syn}:

▶ **Lemma 4.4.** *There are KC^{PS}_{Syn} proofs of PEB_n of size $O(n)$.*

The construction we use is actually very similar to the small kcps^+ proof of PEB_n [3].

Proof. Let $L = (N_{1,1}, N_{2,1}, N_{2,2}, N_{3,1} \dots)$ be list of all nodes in a top-down sorting. We construct a Decision-DNNF D as illustrated in Figure 5. In the root, we take the first node of L and decide its two related variables first. If both variables are true or if both are false, D leads to a 0-gate. Otherwise, if exactly one of both is true, it leads to some fresh gate v. We can label v with $\mathcal{F}_v = \mathsf{PEB}_n[w_{1,1}, \overline{b}_{1,1}]$ because of the symmetry of the pebbling formulas $\mathsf{PEB}_n[w_{1,1}, \overline{b}_{1,1}] = \mathsf{PEB}_n[\overline{w}_{1,1}, b_{1,1}]$. Note that this symmetry holds for more generally, which is essential for this KC^{PS}_{Syn} proof: Let α and β be two complete assignments of all variables belonging to the first k nodes in L such that in both assignments and for every node, exactly one corresponding variable is set to true. Then, $\mathsf{PEB}_n[\alpha] = \mathsf{PEB}_n[\beta]$. Thus, from v, we can do the exact same procedure for the next node in L and continue like that until all nodes are dealt with.

Next, we describe, how we choose the set ρ of resolution refutations that justifies the 0-gates for (vi). For that, assume that v is a 0-gate that sets $w_{i,j} = b_{i,j}$ for some $i, j \in [n]$. If both variables are set to 1, we immediately obtain a conflict to clause $C^9_{i,j}$.

Thus, we only have to consider the case where both variables are assigned to 0, i.e. $(w_{i,j} \vee b_{i,j}) = 0$ meaning that no pebble is placed on node $N_{i,j}$. If we consider the subformula of PEB_n with the pyramidal graph with root $N_{i,j}$, we obtain a formula that is very similar to the formula that is used to separate resolution from tree-like resolution for which the existence of short resolution proofs is known. We construct the resolution proofs completely analogously as follows: We know that the bottom layer has a pebble on every node. With the invariant from clauses $C_1 - C_8$, we derive that every node on the layer above has a pebble placed on it. We continue like that until we conclude that the node $N_{i,j}$ contains a pebble leading to a contradiction.

Note that for this argumentation it is essential that, on the path to v, we do not decide any variable corresponding to a node that has a lower layer than $N_{i,j}$ which is ensured by the ordering of L. The whole derivation can be done with a resolution proof of size linear in the number of nodes to consider. We let ρ be the set of all these resolution derivations for every 0-gate.

We show that the resulting proof $(D, \mathcal{F}, \rho)$ is indeed a valid KC^{PS}_{Syn} proof.

- (i), $D \models \mathsf{PEB}_n$, is satisfied as it is easy to observe that D is equivalent to PEB_n.

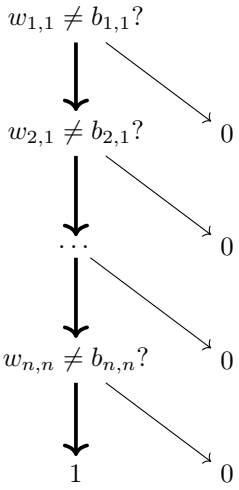

Figure 5 Simplified Decision-DNNF representing PEB_n that can be used for a KC^{PS}_{Syn} proof. The condition $w_{i,j} \neq b_{i,j}$ is a short notation for deciding both variables separately. Thick lines are used if the condition is satisfied.

- (ii), any gate v that is not a 0-gate cannot have assigned $w_{i,j} \neq b_{i,j}$ for any node $N_{i,j}$ with $i, j \in [n]$. Thus, it is still possible to put exactly one pebble on every node and the formula $\mathcal{F}(v)$ is still satisfiable.
- (iii), $\mathcal{F}(r) = \mathsf{PEB}_n$, is true by construction.
- (iv), validity of AND-gates, is satisfied as D does not contain any AND-gate.
- (v), validity of DECISION-gates, is satisfied by construction and the invariant of the PEB_n formulas mentioned above.
- (vi), validity of 0-gates, is satisfied by construction of ρ.

Thus, the KC^{PS}_{Syn} upper bound is shown. ◀

With that, we can show the separation:

Proof of Theorem 4.3. The upper bound for KC^{PS}_{Syn} is shown in Lemma 4.4, while the MICE lower bound is already known [3]. ◀

Therefore, KC^{Alg}_{Syn} with perfectly chosen heuristics is able to handle these separating formulas efficiently while any MICE proof has exponential size.

4.3 Increasing the strength of MICE

Next, we fix this weakness of MICE. As it turns out, this can be done with an additional simple and natural rule. Recall that a MICE claim (F, A, c) states that the formula $F[A]$ is satisfied by exactly the fraction c of all models. Therefore, the following rule seems quite natural: The *Reform* rule allows to derive claim (F_1, A_1) from (F, A) if $F[A]$ and $F_1[A_1]$ are syntactically equivalent formulas. The formal definition of the (Ref) rule is displayed in Figure 6. We refer to MICE with the additional (Ref) rule as MICE_{Ref}. It is easy to verify that MICE_{Ref} is still a valid proof system:

▶ **Proposition 4.5.** MICE_{Ref} *is a sound and complete proof system for #SAT.*

Proof. The completeness of MICE_{Ref} follows directly from the completeness of MICE. To show soundness it is sufficient to show that the (Ref) derivation rules is sound. For that, we have to show that we can only derive correct claims, if we start with a correct one, i.e. let (F_1, A_1, c_1) be a correct claim and we derive (F, A, c_1) from it by applying (Ref). Then, the derived claim is correct, since $c_1 = |\mathsf{Mod}(F[A])| \cdot 2^{-|\mathsf{vars}(F[A])|} = |\mathsf{Mod}(F_1[A_1])| \cdot 2^{-|\mathsf{vars}(F_1[A_1])|}$. Finally, we can verify condition (R-1) efficiently. ◀

Note that for any MICE claim (F, A), we can derive the equivalent claim $(F[A], \emptyset)$ with (Ref) and vice versa.

Reform.

$$\frac{(F_1, A_1, c_1)}{(F, A, c_1)} \quad \text{(Ref)}$$

- (R-1) $F_1[A_1]$ and $F[A]$ are syntactically equivalent. That is, they are the exact same set of clauses after removing weakened clauses.

Figure 6 Additional reform rule for MICE.

Next, we show that this augmented MICE system nicely fits into the bigger picture of #SAT proof systems as it is equivalent to some other system we already know, namely KC^{PS}_{Syn}. Therefore, adding the (Ref) rule avoids the shortcoming of MICE illustrated in Theorem 4.3 and closes the gap between the two systems MICE and KC^{PS}_{Syn}.

▶ **Theorem 4.6.** MICE_{Ref} *is p-equivalent to* KC^{PS}_{Syn}.

We prove the two directions in Lemmas 4.7 and 4.9.

▶ **Lemma 4.7.** MICE_{Ref} *p-simulates* KC^{PS}_{Syn} *with only linear increase of size.*

Proof. Let $\pi = (D, \mathcal{F}, \rho)$ be a KC^{PS}_{Syn} proof of CNF φ and let $v_1, \ldots, v_n$ be a list of all gates in D sorted topologically bottom-up. For every subcircuit D_{v_i}, we define the associated MICE claim $I_i = (\mathcal{F}(v_i), \emptyset)$. We show by induction that we can derive I_k from $I_1, \ldots, I_{k-1}$ efficiently. For that we distinguish, what kind of gate v_k is.

Case 1. v_k is an 1-gate. Then, $\mathcal{F}(v_k)$ has to be a tautology. Since it is also a CNF, it has to be the empty CNF. We can derive $I_k = (\emptyset, \emptyset, 1)$ with (Ax).

Case 2. v_k is a 0-gate. Then, $\mathcal{F}(v_i)$ is unsatisfiable and we can derive I_k with a single application of (Comp) without using other claims. The absence of models statement is the resolution refutation of $\mathcal{F}(v_k)$ which has to be in ρ.

Case 3. v_k is an AND-gate. Let v_i and v_j with $i, j < k$ be its children. By the induction hypothesis we have already derived claims $(\mathcal{F}(v_i), \emptyset)$ and $(\mathcal{F}(v_j), \emptyset)$. By construction of the formulas $\mathcal{F}$ in (iv), $\mathsf{vars}(\mathcal{F}(v_i))$ and $\mathsf{vars}(\mathcal{F}(v_j))$ have to be disjoint, i.e. we can apply (Join) to these two claims which results in $(\mathcal{F}(v_i) \cup \mathcal{F}(v_j), \emptyset) = (\mathcal{F}(v_k), \emptyset)$ by construction of the connected component.

Case 4. v_k is a DECISION-gate that leads with literal l to gate v_i and $\bar{l}$ to v_j with $i, j < k$. By the induction hypothesis we have already derived claims $(\mathcal{F}(v_i), \emptyset)$ and $(\mathcal{F}(v_j), \emptyset)$. By construction $\mathcal{F}(v_k)[l] = \mathcal{F}(v_i)$ and we can derive $(\mathcal{F}(v_k), \{l\})$ with (Ref) from $(\mathcal{F}(v_i), \emptyset)$ and similarly $(\mathcal{F}(v_k), \{\bar{l}\})$ from $(\mathcal{F}(v_j), \emptyset)$. We obtain $I_{v_k} = (\mathcal{F}(v_k), \emptyset)$ by applying (Comp) to these two claims with a trivial absence of model statement that performs a single resolution step over l.

This completes the induction. Since v_n is the root of D, we have derived the claim $I_n = (\mathcal{F}(v_n), \emptyset) = (\varphi, \emptyset)$. The resulting MICE proof π' has a claim for every gate in D and some two additional claims for every decision gate that occurs. Thus, $|\pi'| \leq 3 \cdot |\pi|$. ◀

For the other simulation we first observe that the (Ref) rule is a generalisation of (Ext).

▶ **Observation 4.8.** *Let π be a MICE_{Ref} proof of some formula φ. Then, φ has a MICE_{Ref} proof π' that does not use (Ext) and with $|\pi'| = |\pi|$.*

Further, we want to point out, that Proposition 4.1 naturally also holds for MICE_{Ref}, i.e. we can assume that any MICE_{Ref} proof does only use (0-Ax) and (2-Comp) instead of (Comp).

▶ **Lemma 4.9.** KC^{PS}_{Syn} *p-simulates* MICE_{Ref}.

Proof. Let $\pi = I_1, \ldots, I_n$ with $I_i = (F_i, A_i)$ be a MICE_{Ref} proof of φ. Because of Proposition 4.1, we can assume that π does not use (Comp), but only (0-Ax) and (2-Comp) instead. We inductively construct a circuit D out of labelled nodes n_i, along with resolution derivations ρ_i, so that the subtree of n_i together with ρ_i is a valid KC^{PS}_{Syn} proof of $F_i[A_i]$. We start with $\rho_0 = \emptyset$. We maintain the additional invariant $\mathcal{I}$ that a node deciding a variable x cannot occur at or below a node v with $x \notin \mathsf{vars}(\mathcal{F}_i(v))$.

When adding a node n_k to the circuit, we distinguish how claim I_k is derived.

Case 1. I_k is derived with (Ax), i.e. $I_k = (\emptyset, \emptyset, 1)$. We let n_k be a 1-gate and set $\mathcal{F}(n_k) = \emptyset$, $\rho_k = \rho_{k-1}$. $\mathcal{I}$ is satisfied trivially.

Case 2. I_k is derived with (0-Ax) using some resolution refutation ρ, i.e. $F_k[A_k]$ is unsatisfiable. Then, n_k is a 0-gate. We set $\mathcal{F}(n_k) = F_k[A_k]$ and $\rho_k = \rho_{k-1} \cup \rho$. $\mathcal{I}$ is satisfied trivially.

Case 3. I_k is derived with (2-Comp) using the two claims $I_i = (F_k, A_k \wedge \overline{x})$ and $I_j = (F_k, A_k \wedge x)$ with $i, j < k$. By the induction hypothesis, we have already constructed nodes n_i, n_j representing $F_k[A_k \wedge \overline{x}]$ and $F_k[A_k \wedge x]$. If $x \notin \mathsf{vars}(F_k)$, then $F_k[A_k] = F_k[A_k \wedge x]$ and we can set $n_k = n_j$. This is equivalent to case 5. In the following, we assume that $x \in \mathsf{vars}(F_k)$, satisfying $\mathcal{I}$. We let n_k be a DECISION-gate that leads to n_i if $x = 0$ and to n_j if $x = 1$. Note that this DECISION-gate is indeed allowed in a Decision-DNNF: The variable x cannot be decided later again due to invariant $\mathcal{I}$, because neither $F_k[A_k \wedge x]$ nor $F_k[A_k \wedge \overline{x}]$ contain the variable x. We set $\mathcal{F}(n_k) = F_k[A_k]$. Further, $\rho_k = \rho_{k-1}$.

Case 4. I_k is derived with (Join) using claims $I_i = (F_i, A_i)$ and $I_j = (F_j, A_j)$ with $i, j < k$. By the induction hypothesis, we have already constructed nodes n_i, n_j representing $F_i[A_i]$ and $F_j[A_j]$. We let n_k be an AND-gate with the two children n_i and n_j. First, we show that this AND-gate is indeed decomposable: By (J-2) of (Join), we have $\mathsf{vars}(F_i) \cap \mathsf{vars}(F_j) \subseteq \mathsf{vars}(A_i) \cap \mathsf{vars}(A_j)$ and therefore, $\mathsf{vars}(F_i[A_i]) \cap \mathsf{vars}(F_j[A_j]) = \emptyset$. Since $\mathsf{vars}(D_{n_i}) \subseteq \mathsf{vars}(F_i[A_i])$ and $\mathsf{vars}(D_{n_j}) \subseteq \mathsf{vars}(F_j[A_j])$, we obtain $\mathsf{vars}(D_{n_i}) \cap \mathsf{vars}(D_{n_j}) = \emptyset$.

As in the previous case, we set $\mathcal{F}(n_k) = F_k[A_k]$ and $\rho_k = \rho_{k-1}$. The invariant $\mathcal{I}$ is unchanged.

Case 5. I_k is derived with (Ref) from $I_i = (F_i, A_i)$ with $i < k$. By the induction hypothesis, we have already constructed n_i. Since $F_k[A_k] = F_i[A_i]$, we can simply set $n_k = n_i$. $\mathcal{I}$ is unchanged.

This completes the induction, as $I_n = (\varphi, \emptyset)$, so $\pi' = (D, \mathcal{F}, \rho_n)$ is a KC^{PS}_{Syn} proof of φ. D has at most one gate for every MICE claim, i.e. $|\pi'| \leq |\pi|$. ◀

5 Semantic caching

With a closer look at algorithm KC^{Alg}_{Syn} a natural optimisation springs to mind: improve caching from a purely syntactic approach to a more semantic one. Although it is not obvious

how to do that efficiently in practice, we will investigate this question from a theoretical point of view. That is, we can increase the strength of both KC^{PS}_{Syn} and MICE_{Ref} very naturally if we allow semantic caching and reforming instead of requiring syntactic equivalence of the formulas involved. However, we have to provide some kind of proof for this equivalence.

We start with the formal definitions of the systems.

▶ **Definition 5.1** (KC^{PS}_{Sem}). KC^{PS}_{Sem} *is a variant of* KC^{PS}_{Syn} *where conditions (iii), (iv) and (v) are relaxed: syntactic equality of formulas is replaced by semantic equivalence. For each case where formulas F and G are equivalent, resolution refutations of $F \wedge \overline{G}$ as well as $G \wedge \overline{F}$ are additionally required.*

Semantic Reform.

$$\frac{(F_1, A_1, c_1)}{(F, A, c_1)} \qquad \text{(Ref-Sem)}$$

- (R-1) The MICE proof contains a proof that $F_1[A_1]$ and $F[A]$ are semantically equivalent. That is, there is a resolution refutation of $F_1[A_1 \wedge \overline{C}]$ for every $C \in F[A]$ and of $F[A \wedge \overline{C}_1]$ for every $C_1 \in F_1[A_1]$.

Figure 7 Additional semantic reform rule for MICE.

Before investigating KC^{PS}_{Sem}, we argue that it is indeed a proof system:

▶ **Proposition 5.2.** $\mathsf{MICE}_{Ref\text{-}Sem}$ *is a sound and complete proof system for #SAT.*

Proof. The proof is analogously to the proof that KC^{PS}_{Syn} is a proof system in Proposition 4.5. The only difference is that in the syntactic version, the check $F_1[A_1] = F[A]$ was trivial. To verify that these two formulas are semantically equivalent is a hard problem in general. However, the provided resolution refutations for (R-1) proving exactly this invariant. Thus, we only have to verify that all required resolution refutations are given and correct. ◀

Algorithm KC^{Alg}_{Sem} is defined similarly to KC^{Alg}_{Syn}. The main difference is that the cache lookup in Line 10 uses a semantic cache that can also return entries that are different from, but semantically equivalent to F. Further, a formula F is considered decomposable if it is *semantically* equivalent to $\bigwedge_{i=1}^{k} F_i$ with $k \geq 2$ and variable-disjoint formulas F_i, which is checked by line 12.

How to implement such a semantic cache efficiently is non-trivial and beyond the scope of this paper. However, we assume that such an implementation would use a heuristic to decide whether to check two formulas for equivalence using a SAT solver.

In the non-deterministic algorithm KC^{Alg}_{Sem} we can therefore assume that the only costs are for successful cache lookups. When looking up F and finding G, this cost is the length of resolution refutations of both $F \wedge \overline{G}$ and $G \wedge \overline{F}$.

▶ **Theorem 5.3.** KC^{Alg}_{Sem} *is characterised by* KC^{PS}_{Sem} *(in the same formal sense as in Theorem 3.7).*

Proof. This is nearly equivalent to Theorem 3.7. For extracting a KC^{PS}_{Sem} proof from a solver run, we observe that Theorem 3.2 and Lemmas 3.12 and 3.13 work equivalently in the semantic case. We first extract a C-dec-DNNF proof π according to Theorem 3.2. We use Lemma 3.13, to see that π is formula-decomposable, and use Lemma 3.12 to construct a KC^{PS}_{Sem} proof π' efficiently.

For generating solver runs from proofs, the solver goes through the proof from the top. On DECISION-gates, it branches on the corresponding variable, and on AND-gates it decomposes the formula. If it hits an already visited gate, it uses a semantic cache lookup. If the formula F is changed to a semantically equivalent version G after a DECISION-gate, the solver does not mirror that change and continues with the old version until the next AND-gate or 0-gate. At this point, the solver has a formula $F[\alpha]$ and the proof has $G[\alpha]$, where α is the partial assignment from the intermediate decision nodes. A resolution proof of $F[\alpha] \equiv G[\alpha]$ can be obtained by restricting the resolution proof of $F \equiv G$. At an AND-gate, since $G[\alpha]$ is decomposable, $F[\alpha]$ must also be semantically decomposable. At a 0-gate, $F[\alpha] \equiv G[\alpha]$ and $G[\alpha] \models \bot$ can be combined into a refutation of $F[\alpha]$. ◀

We now strengthen MICE and define $\mathsf{MICE}_{\textit{Ref-Sem}}$ as MICE augmented by the rule (Ref-Sem) in Figure 7. These semantic variants are natural generalisations of KC^{PS}_{Syn} and $\mathsf{MICE}_{\textit{Ref}}$. As in Theorem 4.6, the two systems are of equal strength despite their different approaches:

▶ **Theorem 5.4.** $\mathsf{MICE}_{\textit{Ref-Sem}}$ *and* KC^{PS}_{Sem} *are p-equivalent.*

Proof. Both simulations are very similar to the corresponding one of the syntactic variants in Lemmas 4.7 and 4.9. Proposition 4.1 holds equivalently in the semantic case.

For the direction that KC^{PS}_{Sem} p-simulates $\mathsf{MICE}_{\textit{Ref-Sem}}$, we assume w.l.o.g. that no two consecutive (Ref-Sem) steps occur in the $\mathsf{MICE}_{\textit{Ref-Sem}}$ proof. Each reform step from (F_1, A_1) to (F_2, A_2) is not represented in the Decision-DNNF. Instead, the nodes on either side are connected directly, and the proof that $F_1[A_1] \equiv F_2[A_2]$ is used to justify the change of formula.

For the other simulation, $\mathsf{MICE}_{\textit{Ref-Sem}}$ p-simulates KC^{PS}_{Sem}, we only need to handle the case when formulas are semantically, but not syntactically, equivalent. In this case, we add a (Ref-Sem) step that converts one representation into the other. The required resolution proof for this semantical equivalence has to be provided in the KC^{PS}_{Sem} proof as well and we can just copy it. ◀

To put some perspective on the strength of KC^{PS}_{Sem} we compare it to the proof system $\mathsf{CPOG}^{\mathsf{Decision\text{-}DNNF}}$ [6, 3]:

▶ **Definition 5.5** (Bryant et al. [6], Beyersdorff et al. [3])**.** *A* $\mathsf{CPOG}^{\mathsf{Decision\text{-}DNNF}}$ *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 $\varphi \wedge \mathcal{E}(D) \wedge (\overline{\vartheta_R})$.*

Conceptually, a $\mathsf{CPOG}^{\mathsf{Decision\text{-}DNNF}}$ proof for a CNF φ consists of a Decision-DNNF D together with a resolution proof showing that $\varphi \models D$. The other entailment $D \models \varphi$ is easy to check by Theorem 2.1. In the following two propositions we show that KC^{PS}_{Sem} is almost as strong $\mathsf{CPOG}^{\mathsf{Decision\text{-}DNNF}}$.

▶ **Proposition 5.6.** $\mathsf{CPOG}^{\mathsf{Decision\text{-}DNNF}}$ *p-simulates* KC^{PS}_{Sem}.

Proof. Let $\pi = (D, \mathcal{F}, \rho)$ be a KC^{PS}_{Sem} proof of some CNF φ. For the proof of the lemma we construct a $\mathsf{CPOG}^{\mathsf{Decision\text{-}DNNF}}$ proof π_{CPOG} with the exact same Decision-DNNF D and its straightforward encoding $\mathcal{E}(D)$.

Thus, we only have to extract a resolution refutation of $\varphi \wedge \mathcal{E}(D) \wedge (\overline{v}_R)$ from π. Let $v_1, v_2, \ldots, v_n$ be a bottom-up listing of all gates in D. We inductively build resolution refutations ψ_i for $\mathcal{F}(v_i) \wedge \mathcal{E}(D_{v_i}) \wedge (\overline{v}_i)$. For that, we distinguish what kind of a gate v_i is.

Case 1. v_i is an 1-gate. Then, $\mathcal{E}(D_{v_i})$ contains the clause (v_i), and ψ_i can do a single resolution step together with the clause $(\overline{v}_i)$.

Case 2. v_i is a 0-gate. By definition of the KC^{PS}_{Sem} proof system, ρ has to contain a resolution refutation of $\mathcal{F}(v_i)$, which we can use as ψ_i.

Case 3. v_i is a DECISION-gate which leads to v_s if $x = 0$ and v_t if $x = 1$ with $s, t < i$. We first consider the situation for $x = 0$. $\mathcal{E}(D_{v_i})$ has to encode that if $\overline{x}$, then $v_s = v_i$. For that, it contains the clauses $(x \vee \overline{v}_s \vee v_i)$ and $(x \vee v_s \vee \overline{v}_i)$. By the induction hypothesis, we have already a resolution refutation ψ_s of

$$\mathcal{F}(v_s) \wedge \mathcal{E}(D_{v_s}) \wedge (\overline{v}_s).$$

However, we know that $\mathcal{F}(v_s) \equiv \mathcal{F}(v_i)[\overline{x}]$ by rule (III), and the proof of equivalence contains resolution refutations for $\mathcal{F}(v_i)[\overline{x}] \wedge \overline{C}$ for every clause $C \in \mathcal{F}(v_s)$. By weakening each of these by C, we obtain a derivation of $\mathcal{F}(v_s)$ from $\mathcal{F}(v_i)[\overline{x}]$ that we can combine with the earlier refutation to obtain a refutation of

$$\mathcal{F}(v_i)[\overline{x}] \wedge \mathcal{E}(D_{v_s}) \wedge (\overline{v}_s) = \mathcal{F}(v_i)[\overline{x}] \wedge \mathcal{E}(D_{v_s}) \wedge (x \vee \overline{v}_s)[\overline{x}].$$

If we remove the restrictions $\overline{x}$ in this refutation, we can use it to derive the clause (x):

$$\mathcal{F}(v_i) \wedge \mathcal{E}(D_{v_s}) \wedge (x \vee \overline{v}_s) \vdash (x).$$

As we can derive $(x \vee \overline{v}_s)$ with a single resolution step over $(x \vee \overline{v}_s \vee v_i)$ and $(\overline{v}_i)$, and since $\mathcal{E}(D_{v_s}) \subseteq \mathcal{E}(D_{v_i})$, we can derive

$$\mathcal{F}(v_i) \wedge \mathcal{E}(D_{v_i}) \wedge (\overline{v}_i) \vdash (x).$$

With the analogous procedure, we can also derive $(\overline{x})$ using ψ_t. Finally, we do a resolution step over (x) and $(\overline{x})$ to obtain the empty clause. The whole resulting refutation ψ_i has size $|\psi_i| = |\psi_s| + |\psi_t| + 3$.

Case 4. v_i is an AND-gate with children v_s and v_t with $s, t < i$. By definition, we have $\mathcal{E}(D_{v_i}) = \mathcal{E}(D_{v_s}) \cup \mathcal{E}(D_{v_t}) \cup \{(\overline{v}_i \vee v_s), (\overline{v}_i \vee v_t), (v_i \vee \overline{v}_s \vee \overline{v}_t)\}$. ψ_i has to refute $\mathcal{F}(v_i) \wedge \mathcal{E}(D_{v_i}) \wedge (\overline{v}_i)$. However, we know that $\mathcal{F}(v_i) \equiv \mathcal{F}(v_s) \wedge \mathcal{F}(v_t)$ by rule (III), and the proof of equivalence contains resolution refutations for $\mathcal{F}(v_s) \wedge \mathcal{F}(v_t) \wedge \overline{C}$ for every clause $C \in \mathcal{F}(v_i)$. By weakening each of these by C, we obtain a derivation of $\mathcal{F}(v_r)$ from $\mathcal{F}(v_i)[\overline{x}]$ so we can derive

$$\begin{aligned}
&\mathcal{F}(v_i) \wedge \mathcal{E}(D_{v_i}) \wedge (\overline{v}_i) \\
&\quad \vdash \mathcal{F}(v_s) \wedge \mathcal{F}(v_t) \wedge \mathcal{E}(D_{v_i}) \wedge (\overline{v}_i) \\
&\quad = \mathcal{F}(v_s) \wedge \mathcal{F}(v_t) \wedge \mathcal{E}(D_{v_s}) \wedge \mathcal{E}(D_{v_t}) \wedge (\overline{v}_i \vee v_s) \wedge (\overline{v}_i \vee v_t) \wedge (v_i \vee \overline{v}_s \vee \overline{v}_t) \wedge (\overline{v}_i).
\end{aligned}$$

By removing some unnecessary clauses and do a resolution step over the last two clauses, we can derive

$$\mathcal{F}(v_i) \wedge \mathcal{E}(D_{v_i}) \wedge (\overline{v}_i) \vdash \mathcal{F}(v_s) \wedge \mathcal{F}(v_t) \wedge \mathcal{E}(D_{v_s}) \wedge \mathcal{E}(D_{v_t}) \wedge (\overline{v}_s \vee \overline{v}_t).$$

By the induction hypothesis, we have ψ_s refuting $\mathcal{F}(v_s) \wedge \mathcal{E}(D_{v_s}) \wedge (\overline{v}_s)$ and ψ_t refuting $\mathcal{F}(v_t) \wedge \mathcal{E}(D_{v_t}) \wedge (\overline{v}_t)$. Thus, we can apply ψ_s where we use the clause $(\overline{v}_s \vee \overline{v}_t)$ instead of $(\overline{v}_s)$ to derive $(\overline{v}_t)$ instead of the empty clause. Finally, we can apply ψ_t to derive the empty clause. The whole resulting refutation ψ_i has size $|\psi_i| = |\psi_s| + |\psi_t| + 1$. ◀

The other direction remains open. The main problem seems to be that $\mathsf{CPOG}^{\mathsf{Decision\text{-}DNNF}}$ can introduce helper variables that could potentially shorten proofs. For instance, it is easy to show that PHP is hard for KC^{PS}_{Sem}, but this appears to be a non-trivial result for $\mathsf{CPOG}^{\mathsf{Decision\text{-}DNNF}}$. However, if we replace the underlying proof system resolution by extended resolution, we can prove the simulation.

In extended resolution (ER) we add an additional rule to resolution that allows us to introduce fresh extension variables abbreviating arbitrary formulas [30, 13]. Its strength is illustrated by the fact that ER is polynomially equivalent to very powerful systems like extended Frege [21]. In particular, no exponential lower bounds are known for ER.

▶ **Proposition 5.7.** KC^{PS}_{Sem} *with extended resolution p-simulates* $\mathsf{CPOG}^{\mathsf{Decision\text{-}DNNF}}$.

Proof. Let $(\mathcal{E}(D), \rho)$ be a $\mathsf{CPOG}^{\mathsf{Decision\text{-}DNNF}}$ proof of a CNF φ. We call a gate $v \in D$ *satisfiable* if D_v is satisfiable. We first take the Decision-DNNF D and transform it into an equivalent Decision-DNNF D' where every inner gate is satisfiable, by replacing each unsatisfiable inner gate with a 0-gate. We will use this Decision-DNNF D' and build a formula $F(g)$ for each gate g. We will also build the necessary ER proofs for the equivalence of formulas, as well as ER refutations for $F(g)$ if g is a 0-gate.

For every non-root gate g in D' we arbitrarily pick one ancestor $a(g)$. We recursively define $F(g)$ and $F_b(g)$ for any ancestor b of g, as follows:

- For the root r, $F(r) = \varphi$.
- For any other gate g, $F(g) = F_{a(g)}(g)$.
- If b is a DECISION-gate that leads to g on $x = b$, then $F_b(g) = F(b)[x = b]$.
- If b is an AND-gate with other child h, because it is an inner node, it has to be satisfiable, so g and h are satisfiable. Pick a satisfying assignment α of h and extend it arbitrarily on $\mathsf{vars}(F(b)) \setminus \mathsf{vars}(D_g)$. Set $F_b(g) = F(b)[\alpha]$; this way it only contains variables in $\mathsf{vars}(D_g)$.

This recursively guarantees that $F_b(g) \equiv D_g$. Next, we will build proofs for this. To prove that $F(g) \to D_g$, we will build an ER refutation of $F(g) \wedge \mathcal{E}(D_g) \wedge \overline{g}$. (The other direction, $D_g \to F(g)$, is trivial and always has a short resolution proof.)

We recursively build these refutations in a top-down fashion. For the root gate, the refutation is already part of the $\mathsf{CPOG}^{\mathsf{Decision\text{-}DNNF}}$ proof. For a non-root gate g in D' with ancestor a and sibling h, we already have a refutation of $F(a) \wedge \mathcal{E}(D_a) \wedge \overline{a}$.

If a is a DECISION-gate leading to g if the literal x is satisfied, then $\mathcal{E}(D_a) = \mathcal{E}(D_g) \wedge \mathcal{E}(D_h) \wedge (\overline{x} \vee g \vee \overline{a}) \wedge (\overline{x} \vee \overline{g} \vee a) \wedge (x \vee h \vee \overline{a}) \wedge (x \vee \overline{h} \vee a)$. We restrict the refutation to $x \wedge \overline{a}$ and obtain a refutation of $F(a)[x] \wedge \mathcal{E}(D_a)[x, \overline{a}] = F_a(g) \wedge \mathcal{E}(D_g) \wedge \overline{g} \wedge \mathcal{E}(D_h)$. By introducing the variables of $\mathcal{E}(D_h)$ as auxiliary variables, this can be turned into an ER refutation of $F_a(g) \wedge \mathcal{E}(D_g) \wedge \overline{g}$.

If a is an AND-gate, then $\mathcal{E}(D_a) = \mathcal{E}(D_g) \wedge \mathcal{E}(D_h) \wedge (a \vee \overline{g} \vee \overline{h}) \wedge (\overline{a} \vee g) \wedge (\overline{a} \vee h)$. We reuse the assignment α satisfying D_h that was used to define $F_a(g)$, so that $F_a(g) = F(a)[\alpha]$. Let β be the assignment to the variables in D_h, setting each auxiliary variable is set to the value implied by α. Notably, $\beta(h) = 1$, and $\mathsf{vars}(\beta) \cup \mathsf{vars}(D_g) = \emptyset$. We restrict the refutation to $\alpha \wedge \beta \wedge \overline{a}$ and obtain a refutation of $F(a)[\alpha] \wedge \mathcal{E}(D_a)[\alpha, \beta, \overline{a}] = F_a(g) \wedge \mathcal{E}(D_g)[\alpha, \beta, \overline{a}] \wedge \overline{g} \wedge \mathcal{E}(D_h)[\alpha, \beta] = F_a(g) \wedge \mathcal{E}(D_g) \wedge \overline{g}$.

Using these proofs, we can build a proof of equivalence for each $F(g)$ and $F_a(g)$. We focus on $F(g) \to F_a(g)$, the other direction works analogously. From above, we have ER refutations of $F(g) \wedge \mathcal{E}(D_g) \wedge \overline{g}$, as well as of $\overline{C} \wedge \mathcal{E}(D_g) \wedge g$ for every $C \in F_a(g)$. The first can be turned into a derivation of $F(g) \wedge \mathcal{E}(D_g) \vdash g$. We introduce $\mathcal{E}(D_g)$ as auxiliary variables and combine it with one of the other refutations to obtain an ER refutation of $F(g) \wedge \overline{C}$ for every $C \in F_a(g)$, which is precisely a proof of $F(g) \to F_a(g)$.

Finally, if g' is a 0-gate in G' then the corresponding gate g in G is unsatisfiable. We have a refutation of $F(g) \wedge \mathcal{E}(D_g) \wedge \overline{g}$. However, since g is unsatisfiable, there is a trivial resolution derivation of $\mathcal{E}(D_g) \vdash \overline{g}$: a gate is unsatisfiable only it is either a 0-gate, a DECISION-gate with two unsatisfiable children, or an AND-gate with at least one unsatisfiable child. The resolution derivation can be built trivially from the bottom-up. Using this, we obtain an ER refutation of $F(g) \wedge \mathcal{E}(D_g)$. By adding $\mathcal{E}(D_g)$ as auxiliary variables we obtain an ER refutation of $F(g)$. ◀

Thus, for a CNF φ and any Decision-DNNF $D \equiv \varphi$, we can use D for a KC^{PS}_{Sem} proof, i.e. KC^{PS}_{Sem} is flexible enough to work with arbitrary Decision-DNNFs, a result that does not necessarily hold for the weaker system KC^{PS}_{Syn}.

6 Relations to other #SAT proof systems

So far, we investigated how the new proof systems KC^{PS}_{Syn} and KC^{PS}_{Sem} are related to MICE and $\mathsf{CPOG}^{\text{Decision-DNNF}}$. In this section want to provide the bigger picture of all #SAT proof systems that exist so far. The resulting simulation order is given in Figure 8 for which we provide some more details in the remainder of this section.

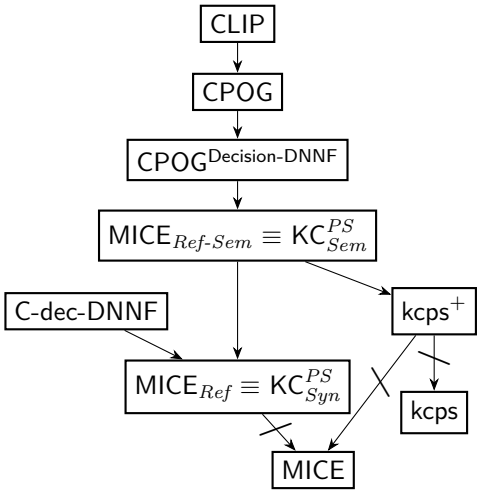

Figure 8 Detailed simulation order of all current #SAT proof systems. A solid edge from A to B indicates that A p-simulates B. If the edge is crossed, A is also exponentially separated from B.

We give some intuition and the definitions of the proof systems kcps and its improved version kcps^+ [8, 3]. The main idea is similar to $\mathsf{CPOG}^{\text{Decision-DNNF}}$: We want to provide a Decision-DNNF D that represents exactly the CNF φ we want to count on. While $D \models \varphi$ can always be checked efficiently, $\varphi \models D$ is in general NP-hard. Thus, kcps and kcps^+ use some restricted and annotated Decision-DNNFs where this check becomes easy.

For the formal definition, let S be a set of clauses. We call a Decision-DNNF D *S-certified* [8] if every 0-gate $N \in D$ is labelled by a *certificate* $C \in S$. Here, a clause is a certificate for N if all assignments that reach N falsify C.

▶ **Definition 6.1** ([8]). *A* kcps *proof of a CNF φ is a φ-certified Decision-DNNF D where $D \equiv \varphi$.*

▶ **Definition 6.2** ([8, 3]). *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 $D \equiv \varphi$.*

As C-dec-DNNF can be seen as an improved and stronger version of these systems, the following result is not surprising:

▶ **Proposition 6.3.** KC^{PS}_{Sem} *p-simulates* kcps^+.

Proof. Let $\pi = (\sigma, D)$ be a kcps^+ proof for a CNF φ. We construct a KC^{PS}_{Sem} proof from π that uses the same Decision-DNNF D. We put an arbitrary spanning tree T on D and define CNFs $\mathcal{F}$ in a top-down fashion according to T starting with $\mathcal{F}(r) = \varphi$. By the correctness of the kcps^+ proof system we can w.l.o.g. assume that any node $v \in D$ satisfies $D_v \equiv \mathcal{F}(v)$.

Further, we assume w.l.o.g. that for any inner gate $v \in D$, $\mathcal{F}(v)$ is satisfiable. If this would not be the case, $\mathcal{F}(v)$ would be unsatisfiable and D_v would be a kcps^+ proof that $\mathcal{F}(v)$ is unsatisfiable. However, it is known that, on unsatisfiable formulas, kcps^+ is p-equivalent to resolution [3]. Thus, we can replace v by a 0-gate and can also efficiently derive a clause that can be used as annotation for this 0-gate.

With that, condition (i) is satisfied as $D \equiv \varphi$ by correctness of kcps^+. (ii), inner gates are satisfiable, is satisfied by the construction of D above. (iii), $\mathcal{F}(r) = \varphi$, is satisfied by the construction of $\mathcal{F}$.

For (vi), we argue how to build the resolution refutations for the 0-gates: Let $v \in D$ be 0-gate and let α_v be the unique path from the root to v according to T. Further, let C_v be the clause that v is annotated with and let ρ_v be the resolution derivation $\varphi \vdash C_v$ in σ. By restricting ρ_v, we obtain a derivation $\varphi[\alpha_v] \vdash C_v[\alpha_v] = \square$ as α_v falsifies C_v by the definition of a kcps^+ proof. With the same argumentation from the proof of Lemma 3.12, we have $\varphi[\alpha_v] = \mathcal{F}(v) \cup X_v$ for some formula X_v with $\mathsf{vars}(\mathcal{F}(v)) \cap \mathsf{vars}(X_v) = \square$ which is formed by previous AND-gates. Also, as in the other proof, there has to exist some assignment α_{X_v} that satisfies X_v such that $\rho_v[\alpha_v \wedge \alpha_{X_v}]$ is a derivation $\mathcal{F}(v) \vdash \square$. Let the set ρ of resolution refutations for (vi) contain all these restricted refutations $\rho_v[\alpha_v \wedge \alpha_{X_v}]$ for every 0-gate $v \in D$.

Finally, for (iv) and (v), we have to construct the resolution refutations to show the semantical equivalences. For that, consider some gate $v \in D$ and let P_v be the canonical path from the root r to v according to T and let P'_v be a different path from r to v. By construction, if we compute $\mathcal{F}$ according to P, we obtain $\mathcal{F}(v)$. Let $F_{P'_v}$ be the formula that would result, if we would have computed $\mathcal{F}$ according to P'_v instead. Because of $\mathcal{F}(v) \equiv D_v \equiv F_{P'_v}$, $\mathcal{F}(v)$ and $F_{P'_v}$ are indeed semantically equivalent. In fact, we can show this equivalence efficiently with resolution by exploiting Theorem 2.1. This theorem states that we can verify both $D_v \models F_{P'_v}$ and $D_v \models \mathcal{F}(v)$ efficiently and it is not too difficult to observe that this can be done in resolution efficiently as well. These two statements together ensure $\mathcal{F}(v) \equiv F_{P'_v}$.

Additionally, for (iv), we have to make sure that for any AND-gate v with children s and t, we have to make sure that $\mathsf{vars}(\mathcal{F}(s)) \cap \mathsf{vars}(\mathcal{F}(t)) = \emptyset$. This follows from the fact that v has to be decomposable, i.e. $\mathsf{vars}(D_s) \cap \mathsf{vars}(D_t) = \emptyset$ and $\mathcal{F}(s) \equiv D_s$, $\mathcal{F}(t) \equiv D_t$.

With that, $(D, \mathcal{F}, \rho)$ is indeed a valid KC^{PS}_{Sem} proof for φ. ◀

Next, we give a brief introduction to CPOG [6] which is a more powerful variant of $\mathsf{CPOG}^{\mathsf{Decision\text{-}DNNF}}$ where we do not restrict ourselves to Decision-DNNFs but allow *Partitioned-Operation Graphs* (POGs) that are circuits with decomposable AND-gates, deterministic OR-gates and NOT-gates. That is, POGs seems to be the most powerful generalisation of Decision-DNNFs such that we can still count on them efficiently.

▶ **Definition 6.4.** *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.*

Finally, the strongest system for #SAT we have so far is CLIP. While all other systems are motivated from a practical point of view, CLIP is a system for purely theoretical interests. For a formal definition, we refer to Chede, Chew, and Shukla [11], where it is also shown that CLIP p-simulates CPOG which we used for Figures 8.

With that, we obtain the full picture of the current proof systems for #SAT displayed in Figure 8.

7 Conclusion

While Figure 8 from the previous section gives an overview of the whole landscape of #SAT proof system, we want to conclude by giving some perspectives on what, in our opinion, the most relevant proof systems are.

We advocate the view that the right formulation of MICE is to include the (Ref) rule, which was simply missing in the original definition. This is confirmed by (1) the fact that MICE_{Ref} is equivalent to KC^{PS}_{Syn}, thus leading to a very robust characterisation of the strength of this proof system and (2) its equivalence to state-of-the-art model counting, which was the precise original motivation for the introduction of MICE [18].

All of kcps, kcps^+ and C-dec-DNNF were suggested as #SAT proof systems close to solvers. In view of Theorem 3.7, we believe that KC^{PS}_{Syn} (which is conceptually close to all the three systems mentioned) is the right answer, further confirmed by the equivalence to MICE_{Ref}. Aesthetically, however, kcps^+ is more pleasing as its definition is simpler, hence kcps^+ could have theoretical interest.

Further, KC^{PS}_{Sem} is *almost* as strong as $\mathsf{CPOG}^{\mathsf{Decision\text{-}DNNF}}$. For both systems, we can use arbitrary Decision-DNNFs to represent the formula on which we count. Should $\mathsf{CPOG}^{\mathsf{Decision\text{-}DNNF}}$ be stronger, then only for propositional reasons, because when changing the underlying proof system from resolution to extended resolution, they become equivalent. Thus, the two systems appear to be quite close to each other (with KC^{PS}_{Sem} potentially being the better choice).

Finally, CLIP [11] is a theoretically elegant, but very powerful proof system, far beyond current solving techniques or proof logging needs.

Thus, in our view, the emerging picture of the most relevant #SAT proof systems together with their potential for solving can be displayed as in Figure 9.

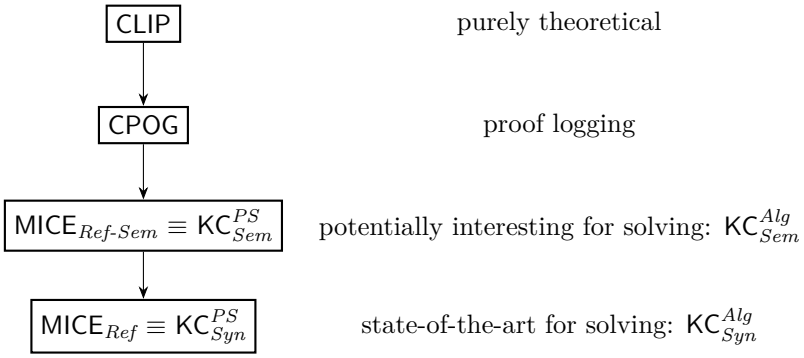

Figure 9 The most relevant proof systems for #SAT (left) and their usage/potential for solving (right).

All simulations from Figure 9 are proven, and we also conjecture that all systems can in fact be separated.

The separation of KC^{PS}_{Syn} and KC^{PS}_{Sem} appears to be of particular interest, as this would provide insights on whether semantic caching could potentially improve practical model counting.

References

1 Fahiem Bacchus, Shannon Dalmao, and Toniann Pitassi. Algorithms and complexity results for #SAT and Bayesian inference. In *FOCS 2003*, pages 340–351. IEEE Computer Society, 2003.

2 Teodora Baluta, Zheng Leong Chua, Kuldeep S. Meel, and Prateek Saxena. Scalable quantitative verification for deep neural networks. In *ICSE 2021*, pages 312–323. IEEE, 2021.

3 Olaf Beyersdorff, Johannes Klaus Fichte, Markus Hecher, Tim Hoffmann, and Kaspar Kasche. The relative strength of #sat proof systems. In *SAT '24*, volume 305 of *LIPIcs*, pages 5:1–5:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024. URL: `https://doi.org/10.4230/LIPIcs.SAT.2024.5`, `doi:10.4230/LIPICS.SAT.2024.5`.

4 Olaf Beyersdorff, Tim Hoffmann, and Luc Nicolas Spachmann. Proof complexity of propositional model counting. *J. Satisf. Boolean Model. Comput.*, 15(1):27–59, 2024. URL: `https://doi.org/10.3233/sat-231507`, `doi:10.3233/SAT-231507`.

5 Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors. *Handbook of Satisfiability*, Frontiers in Artificial Intelligence and Applications. IOS Press, 2021.

6 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`.

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

8 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`.

9 Florent Capelli, Jean-Marie Lagniez, and Pierre Marquis. Certifying top-down decision-dnnf compilers. In *AAAI '21*, pages 6244–6253, 2021. URL: `https://doi.org/10.1609/aaai.v35i7.16776`, `doi:10.1609/AAAI.V35I7.16776`.

10 Supratik Chakraborty, Kuldeep S. Meel, and Moshe Y. Vardi. Approximate model counting. In *Handbook of Satisfiability - Second Edition*, volume 336 of *Frontiers in Artificial Intelligence and Applications*, pages 1015–1045. IOS Press, 2021. `doi:10.3233/FAIA201010`.

11 Sravanthi Chede, Leroy Chew, and Anil Shukla. Circuits, proofs and propositional model counting. In *FSTTCS '24*, volume 323 of *LIPIcs*, pages 18:1–18:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024. URL: `https://doi.org/10.4230/LIPIcs.FSTTCS.2024.18`, `doi:10.4230/LIPICS.FSTTCS.2024.18`.

12 Stephen A. Cook. The complexity of theorem proving procedures. In *Proc. 3rd Annual ACM Symposium on Theory of Computing*, pages 151–158, 1971.

13 Stephen A. Cook and Robert A. Reckhow. The relative efficiency of propositional proof systems. *The Journal of Symbolic Logic*, 44(1):36–50, 1979.

14 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`.

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

16 Leonardo Dueñas-Osorio, Kuldeep S. Meel, Roger Paredes, and Moshe Y. Vardi. Counting-based reliability estimation for power-transmission grids. In *AAAI 2017*, pages 4488–4494. AAAI Press, 2017.

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

18 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`.

19 Jinbo Huang and Adnan Darwiche. The language of search. *J. Artif. Intell. Res.*, 29:191–219, 2007.

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

21 Jan Krajíček. *Bounded Arithmetic, Propositional Logic, and Complexity Theory*, volume 60 of *Encyclopedia of Mathematics and Its Applications*. Cambridge University Press, Cambridge, 1995.

22 Jean-Marie Lagniez and Pierre Marquis. An improved decision-dnnf compiler. In *IJCAI '17*, pages 667–673. ijcai.org, 2017. URL: `https://doi.org/10.24963/ijcai.2017/93`, `doi:10.24963/IJCAI.2017/93`.

23 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 2017*, volume 10416, pages 495–511. Springer, 2017.

24 João P. Marques Silva, Inês Lynce, and Sharad Malik. Conflict-driven clause learning SAT solvers. In Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors, *Handbook of Satisfiability*, Frontiers in Artificial Intelligence and Applications. IOS Press, 2021.

25 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 of *Lecture Notes in Computer Science*, pages 356–361. Springer, 2012. `doi:10.1007/978-3-642-30353-1\_36`.

26 Knot Pipatsrisawat and Adnan Darwiche. On the power of clause-learning SAT solvers as resolution engines. *Artif. Intell.*, 175(2):512–525, 2011.

27 Weijia Shi, Andy Shih, Adnan Darwiche, and Arthur Choi. On tractable representations of binary neural networks. In *KR 2020*, pages 882–892, 2020.

28 Marc Thurley. sharpsat - counting models with advanced component caching and implicit BCP. In *SAT '06*, volume 4121 of *Lecture Notes in Computer Science*, pages 424–429. Springer, 2006. `doi:10.1007/11814948\_38`.

29 S. Toda. PP is as hard as the polynomial-time hierarchy. *SIAM Journal on Computing*, 20:865–877, 1991.

30 G. C. Tseitin. On the complexity of derivations in propositional calculus. In A. O. Slisenko, editor, *Studies in Mathematics and Mathematical Logic, Part II*, pages 115–125. 1968.

31 Nathan Wetzler, Marijn Heule, and Warren A. Hunt Jr. Drat-trim: Efficient checking and trimming using expressive clausal proofs. In Carsten Sinz and Uwe Egly, editors, *Theory and*

Applications of Satisfiability Testing (SAT, volume 8561 of *Lecture Notes in Computer Science*, pages 422–429. Springer, 2014.

32 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 2020*, pages 575–589. USENIX Association, 2020.

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il