Efficient Minimization of Decision-DNNF Circuits via Semantic Hashing and Provenance Tracking

Armin Biere[1] , **Jean-Marie Lagniez**[2] , **Emmanuel Lonca**[2]
[1]University of Freiburg, Germany
[2]Univ. Artois, CNRS, CRIL, France
{lagniez, lonca}@cril.fr, biere@cs.uni-freiburg.de

Abstract

Knowledge Compilation transforms propositional formulas into tractable structures like decision-DNNF to support efficient reasoning. However, these representations often suffer from exponential size, and standard minimization via SAT sweeping is computationally prohibitive for large instances. In this paper, we propose a scalable minimization framework for decision-DNNF that eliminates the need for SAT solvers. We introduce a semantic hashing technique leveraging polynomial-time model counting to rapidly filter redundancies, followed by a polynomial-time verification strategy based on CNF projection. Our experimental evaluation demonstrates that this approach efficiently compresses decision-DNNF circuits while avoiding the bottleneck of NP-hard equivalence checks.

1 Introduction

Propositional logic serves as a cornerstone of Artificial Intelligence, providing a rigorous approach to knowledge representation and reasoning [Brachman and Levesque, 2004]. It underpins critical AI subfields, including Explainable AI (XAI) [Darwiche, 2023], automated reasoning [Biere *et al.*, 2021], and decision support systems [Plaisted, 2015]. However, expressiveness of propositional logic comes at a computational cost: core inference tasks, such as satisfiability (SAT), are NP-hard [Cook, 1971]. This inherent complexity of SAT solving creates a bottleneck for real-time applications, particularly in high-stakes environments where latency and accuracy are non-negotiable.

Industrial applications exacerbate this challenge, demanding scalability for massive problem instances. For example, AWS's Zelkova engine [Rungta, 2022] processes billions of verification queries daily, highlighting the urgent need for optimized reasoning pipelines [Biere *et al.*, 2021]. To mitigate on-line computational costs, *Knowledge Compilation* (KC) has emerged as a standard paradigm [Cadoli and Donini, 1997; Darwiche and Marquis, 2002]. KC shifts the computational burden to an off-line phase, compiling propositional formulas into target languages that support polynomial-time querying. Prominent among these is the *Decision-DNNF* (decision-DNNF), which renders tasks like Weighted Model Counting (WMC) tractable. The practical utility of this approach is evident in systems like Renault's configuration tool [Astesana *et al.*, 2010], where precomputed compilations enable rapid product customization.

Nevertheless, Knowledge Compilation is not a panacea; it trades time complexity for space complexity. While decision-DNNF circuits facilitate efficient inference, they are susceptible to exponential blow-up in size, which severely hampers scalability and storage efficiency [Bova *et al.*, 2016]. This 'tractability-vs-size' trade-off presents a significant barrier to deploying KC in resource-constrained or large-scale scenarios. Consequently, minimizing these compiled representations is a critical post-processing step. State-of-the-art reduction frameworks, such as those inspired by FRAIGs (Functionally Reduced And-Inverter Graphs), typically rely on SAT sweeping [Biere *et al.*, 2024b; Amarù *et al.*, 2020; Kuehlmann, 2004; Lu *et al.*, 2003]. This process involves incrementally solving a sequence of topologically ordered SAT problems to identify and merge functionally equivalent nodes. However, a major bottleneck of this approach is the sheer volume of solver calls required, which becomes computationally prohibitive for large circuits.

In this paper, we propose a novel minimization framework that avoids expensive SAT calls by exploiting the inherent properties of the target language itself. Our key insight is that since decision-DNNF representations allow for polynomial-time model counting, we can leverage semantic information (specifically model counts) to accelerate equivalence checking. We introduce a specialized hashing mechanism that utilizes the model count and the variable set of each node as a semantic signature. Unlike pure structural hashing, this hybrid approach integrates both combinatorial and numerical properties to efficiently identify candidate sub-circuits for merging. By computing these signatures, we can rapidly cluster structurally similar components and drastically reduce the search space for equivalence testing.

Furthermore, we supplant the traditional SAT-based verification step with a polynomial-time refinement process. We achieve this by tagging the decision-DNNF structure with metadata derived from the original CNF formulas. These CNF-based annotations allow us to trace the provenance of sub-circuits and thus verify logical equivalence through a polynomial structured check, ensuring correctness without reverting to full NP-hard reasoning.

Our contributions can be summarized as follows: we propose a specialized hashing function for decision-DNNF minimization that utilizes model counts and variable sets to detect functional redundancy. Additionally, we introduce a polynomial-time verification strategy leveraging CNF-based annotations, which eliminates the need for SAT sweeping. Finally, we demonstrate that this approach significantly reduces circuit size while strictly preserving all semantic properties.

Section 2 introduces the necessary notation. Section 3 details our minimization framework, combining semantic hashing with polynomial-time verification. We discuss related work in Section 4 and present our experimental results in Section 5. Finally, Section 6 concludes the paper.

2 Formal Preliminaries

We consider a propositional language $\mathcal{L}$ defined over a finite set of variables $\mathcal{X}$. A *literal* is a variable $x \in \mathcal{X}$ or its negation $\neg x$. For a literal ℓ, $Var(\ell)$ denotes the underlying variable. A *formula* Σ is constructed using standard logical connectives ($\wedge$, $\vee$, $\neg$) and the logical constants $\top$ (true) and $\bot$ (false). $Var(\Sigma)$ denotes the set of variables appearing in Σ.

Semantics are defined as usual. An assignment ω is a mapping $\omega : \mathcal{X} \to \{0, 1\}$. We often view an assignment ω as the set of literals it satisfies. Under this interpretation, we can apply standard set-theoretic operations (e.g., intersection, union) to assignments. Given a subset of variables $X \subseteq \mathcal{X}$, $\omega[X]$ denotes the projection of the assignment onto X.

We say ω *satisfies* Σ, denoted $\omega \models \Sigma$, if Σ evaluates to 1 under ω. The set of all satisfying assignments (or models) of Σ is denoted by $Mod(\Sigma)$. If $Mod(\Sigma) = \emptyset$, the formula is *unsatisfiable* (or inconsistent), denoted as $\Sigma \equiv \bot$. Two formulas Σ_1 and Σ_2 are *equivalent*, denoted $\Sigma_1 \equiv \Sigma_2$, iff $Mod(\Sigma_1) = Mod(\Sigma_2)$. Finally, Σ_1 *implies* Σ_2, denoted $\Sigma_1 \models \Sigma_2$, iff $Mod(\Sigma_1) \subseteq Mod(\Sigma_2)$.

A *Conjunctive Normal Form* (CNF) formula is a conjunction of clauses, where each clause is a disjunction of literals. We view a CNF Σ as a set of clauses, and a clause as a set of literals. The conditioning of Σ by a literal ℓ, denoted $\Sigma_{|\ell}$, is obtained by replacing every occurrence of ℓ with $\top$ and every occurrence of its negation $\neg\ell$ with $\bot$. This is followed by standard Boolean simplifications (e.g., removing clauses satisfied by $\top$ and removing $\bot$ from clauses) until the formula is minimal. This notion extends to a set of literals $S = \{\ell_1, \ldots, \ell_m\}$ by iterative application:

$$\Sigma_{|S} = (\ldots(\Sigma_{|\ell_1})\ldots)_{|\ell_m}$$

Example 1. *Consider the following CNF formula Σ with four clauses defined over the variables $\mathcal{X} = \{x_1, \ldots, x_5\}$:*

$$\Sigma = \{\, (\neg x_1 \vee x_2 \vee x_3 \vee x_4), (x_1 \vee x_2 \vee x_3), \\ (\ x_1 \vee x_3 \vee x_4 \vee x_5), (x_1 \vee \neg x_3 \vee x_4 \vee x_5)\,\}$$

Conditioning Σ on x_1 satisfies the last three clauses with x_1 and simplifies the first clause (removing $\neg x_1$), yielding:

$$\Sigma_{|\{x_1\}} = \{(x_2 \vee x_3 \vee x_4)\}$$

We further consider formulas in *deterministic Decomposable Negation Normal Form* (d-DNNF), which are represented by a rooted Directed Acyclic Graph (DAG) where leaves are literals or constants, and internal nodes are either decomposable AND gates or deterministic OR gates [Darwiche, 2002]. Decomposability requires that for any AND node, the variable sets of its children are disjoint. Determinism requires that for any OR node, the children are mutually inconsistent (disjoint models).

In this work, we focus on a specific subset of d-DNNF known as *decision-DNNF* [Huang and Darwiche, 2005] (also referred to as Decomposable Decision Graphs [Fargier and Marquis, 2006]). In a decision-DNNF, the logical structure is defined by *decision nodes*. A decision node N is of the form $\text{ite}(x, N_\top, N_\bot)$, representing "**if** x **then** $N_\top$ **else** $N_\bot$". Semantically, this is equivalent to $(x \wedge N_\top) \vee (\neg x \wedge N_\bot)$. To strictly adhere to the d-DNNF properties we assume:

- **Decomposability:** The decision variable x must not appear in the sub-circuits rooted at $N_\top$ or $N_\bot$.
- **Determinism:** Guaranteed by the structure, as x and $\neg x$ are mutually exclusive.

Modern compilers like **d4** [Lagniez and Marquis, 2017a], **SharpSAT-TD** [Kiesel and Eiter, 2023], **Dsharp** [Muise *et al.*, 2012] and **C2D** [Darwiche, 2004] typically output this format. The size of a decision-DNNF, denoted $|\Sigma|$, is the number of edges in the graph.

To facilitate the presentation of our reduction rules, we assume each node in the decision-DNNF is associated with a unique index i, denoted as N_i. We define the following properties for any given node N_i:

- $Var(N_i)$: The set of variables in the scope of the sub-circuit rooted at N_i.
- $Lits(N_i)$: The set of literals (decisions) encountered on the path from the root to N_i. In cases where N_i appears as a child of both branches of a decision node, the negative literal is systematically selected.
- $circ(N_i)$: The structural sub-circuit rooted at N_i.
- $mc(N_i)$: The model count of $circ(N_i)$, i.e., the number of satisfying assignments over $Var(N_i)$.

A key advantage of decision-DNNFs is that $mc(N_i)$ can be computed in linear time via a bottom-up traversal:

- If N_i is a literal or $\top$, $mc(N_i) = 1$. If N_i is $\bot$, $mc(N_i) = 0$.
- For a decision node $N_i = \text{ite}(x, N_\top, N_\bot)$, the count is the sum of its children: $mc(N_i) = mc(N_\top) + mc(N_\bot)$. (This assumes the circuit is *smooth*, meaning $N_\top$ and $N_\bot$ cover the same set of variables).
- For an AND node $N_i = \wedge(N_{j_1}, \ldots, N_{j_k})$, the count is the product of its children: $mc(N_i) = \prod_{m=1}^{k} mc(N_{j_m})$.

Example 2 (Example 1 cont'ed)**.** *Figure 1 depicts a valid decision-DNNF, denoted Ψ, representing the CNF formula Σ. The size of this circuit is $|\Psi| = 18$. The model count of the root node corresponds to the count of the entire formula: $mc(N_0) = mc(\Sigma) = 23$. We can observe specific properties for internal nodes:*

- ***Variables:*** *Nodes N_3 and N_5 share the same variable scope: $Var(N_3) = Var(N_5) = \{x_2, x_3\}$.*

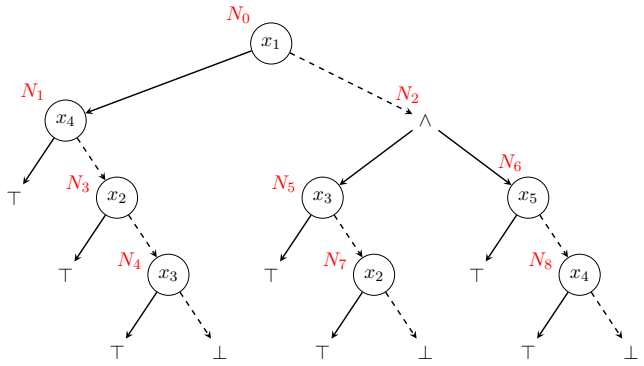

Figure 1: A decision-DNNF corresponding to the CNF formula Σ from Example 1. Circular nodes denote decision steps, where solid edges indicate the high branch ($x = \top$) and dashed edges indicate the low branch ($x = \bot$). The $\wedge$ node represents a decomposable conjunction. Note that terminal nodes ($\top, \bot$) are duplicated in this figure for visual clarity, whereas they are typically shared as unique sinks in implementation.

- ***Model Counts:*** *Nodes N_3 and N_5 exhibit identical model counts: $mc(N_3) = mc(N_5) = 3$.*
- ***Path Literals:*** *Nodes N_3 and N_5 are reached via different paths. N_3 is reached via the high branch of x_1 and low branch of x_4, so $Lits(N_3) = \{x_1, \neg x_4\}$. Conversely, N_5 is reached via the low branch of x_1, so $Lits(N_5) = \{\neg x_1\}$.*

3 Proposed Minimization Framework

Our objective is to minimize a decision-DNNF Ψ by merging functionally equivalent sub-circuits. Let N_i and N_j be two nodes in Ψ. We say $N_i \equiv N_j$ if the Boolean functions represented by the sub-circuits rooted at N_i and N_j are logically equivalent, i.e., $circ(N_i) \equiv circ(N_j)$. A naive approach would perform a quadratic number of equivalence checks using a SAT solver, which is intractable. Instead, we propose a scalable two-phase strategy consisting of *Semantic Hashing*, a constant-time heuristic filter to identify candidate pairs, followed by *CNF-Based Verification*, a polynomial-time determinization step leveraging the original CNF Σ.

3.1 Semantic Hashing

The first phase rapidly filters obvious non-equivalences using a semantic signature. Given the properties of decision-DNNFs defined in Section 2, two equivalent nodes must necessarily share the same variable scope and the same model count. We define the semantic hash of a node N_i as a tuple:

$$H(N_i) = \langle Var(N_i), mc(N_i) \rangle$$

Before identifying candidate pairs, we compute $H(N_i)$ for all nodes in a single bottom-up pass. If $H(N_i) \neq H(N_j)$, then $N_i \not\equiv N_j$. This acts as a powerful pruning metric: matching model counts is a strong indicator of potential equivalence, drastically reducing the search space compared to purely "structural hashing" (*cf.* [Biere *et al.*, 2024a]), which misses semantically equivalent but structurally different graphs.

Example 3 (Example 1 continued)**.** *Revisiting the decision-DNNF depicted in Figure 1, we observe that nodes N_3 and N_5 share identical signatures:*

$$H(N_3) = H(N_5) = \langle \{x_2, x_3\}, 3 \rangle$$

Thus they are identified as a candidate pair for equivalence and grouped into the same bucket for subsequent verification.

3.2 Verification via CNF Projection

While $H(N_i) = H(N_j)$ is a necessary condition for equivalence, it is not sufficient (collisions can occur). To verify equivalence without invoking a SAT solver, we leverage the provenance of the circuit – specifically, the original CNF formula Σ from which Ψ was compiled. The core idea is to associate each node N_i with a specific fragment of Σ that precisely defines the logic of that sub-circuit.

Conditioning by Path Literals. Any node N_i operates within a context defined by the path literals $Lits(N_i)$ from the root. Intuitively, conditioning the original formula Σ by these literals restricts the global function to the specific branch currently traversed. However, a scope mismatch arises: while the sub-circuit N_i is defined strictly over $Var(N_i)$, the conditioned formula $\Sigma|_{Lits(N_i)}$ remains defined over all variables not assigned on the path – including those belonging to parallel branches (siblings). Consequently, $\Sigma|_{Lits(N_i)}$ does not correspond solely to N_i, but rather to the conjunction of N_i and the constraints imposed by all sibling sub-circuits encountered on the path.

Proposition 1 (Conditioned Equivalence)**.** *Let Σ be a CNF formula represented by a decision-DNNF Ψ. For any node N_i, let π be the path from the root to N_i defined by $Lits(N_i)$. We define the set of path-relevant nodes, denoted $\mathcal{K}(\pi)$, as the union of $\{N_i\}$ and the set of all siblings of AND nodes appearing on π. Then:*

$$\Sigma|_{Lits(N_i)} \equiv \bigwedge_{N_j \in \mathcal{K}(\pi)} circ(N_j)$$

Proof. We proceed by induction on the length of the path π from the root N_0 to N_i.

Base Case: The path is empty (length 0), so $N_i = N_0$ (the root). Here, $Lits(N_0) = \emptyset$. The set $\mathcal{K}(\pi) = \{N_0\}$ (no ancestors, so no siblings). The statement reduces to $\Sigma \equiv circ(N_0)$, which holds by the definition of the circuit representing Σ.

Inductive Step: Assume that for a node P with path π_P, we have:

$$\Sigma|_{Lits(P)} \equiv \bigwedge_{N_k \in \mathcal{K}(\pi_P)} circ(N_k)$$

Let C be a child of P. We distinguish two cases depending on the type of P:

Case 1: P is a Decision Node ($P = ite(x, P_\top, P_\bot)$). Assume w.l.o.g. that $C = P_\top$. The path π_C extends π_P with the decision literal x. Thus, $Lits(C) = Lits(P) \cup \{x\}$. The set of relevant nodes remains unchanged except replacing P with C (no AND siblings added), so conceptually $\mathcal{K}(\pi_C) \equiv \mathcal{K}(\pi_P) \setminus \{P\} \cup \{C\}$. From the hypothesis: $\Sigma|_{Lits(P)} \equiv P \wedge \Phi_{rest}$, where Φ_{rest} are the ancestor siblings. Conditioning on x:

$$\Sigma|_{Lits(C)} = (\Sigma|_{Lits(P)})|_x \equiv (P \wedge \Phi_{rest})|_x$$

Since $P|_x \equiv P_\top = C$ (by definition of ITE), and Φ_{rest} does not depend on x, we get $\Sigma|_{Lits(C)} \equiv C \wedge \Phi_{rest}$. This matches the conjunction over $\mathcal{K}(\pi_C)$.

Case 2: P is a Decomposable AND Node ($P = C \wedge S$). Here, C is the child we traverse, and S is the sibling. The path π_C extends π_P but adds no new literals (AND nodes are not decisions), so $Lits(C) = Lits(P)$. However, the structure changes: P decomposes into $C \wedge S$. The set $\mathcal{K}(\pi_C)$ now includes the new sibling S. From the hypothesis:

$$\Sigma|_{Lits(C)} = \Sigma|_{Lits(P)} \equiv P \wedge \Phi_{rest}$$

Substituting $P \equiv C \wedge S$:

$$\Sigma|_{Lits(C)} \equiv (C \wedge S) \wedge \Phi_{rest}$$

This is exactly the conjunction of the current node (C), the new sibling (S), and previous siblings (Φ_{rest}), which corresponds to $\bigwedge_{N_j \in \mathcal{K}(\pi_C)} circ(N_j)$.

Conclusion: By induction, the proposition holds for all nodes reachable from the root. □

Example 4 (Example 1 continued)**.** *Revisiting the decision-DNNF depicted in Figure 1, consider the path defined by $\neg x_1$, we have: $\Sigma|_{\neg x_1} \equiv circ(N_5) \wedge circ(N_6)$.*

Due to the decomposable nature of AND nodes in Ψ, variables outside $Var(N_i)$ typically belong to sibling sub-circuits and should not influence the equivalence of $circ(N_i)$. In the following, we show how to pinpoint the exact CNF fragment equivalent to $circ(N_i)$.

Clause Separation Property. To isolate the logic of N_i, we must filter $\Sigma|_{Lits(N_i)}$ to retain only clauses relevant to $Var(N_i)$. We rely on the following property of decomposable representations:

Proposition 2 (Decomposable Clause Separation)**.** *Let Σ be a CNF formula representing a function f. Assume f decomposes into $f(A) \wedge f(B)$ where A and B are disjoint sets of variables. If clause $\alpha \in \Sigma$ can be written as $\alpha = \beta_1 \vee \beta_2$ with $Var(\beta_1) \subseteq A$ and $Var(\beta_2) \subseteq B$, then $\Sigma \models \beta_1$ or $\Sigma \models \beta_2$.*

Proof. First, if Σ is unsatisfiable, the proposition holds vacuously. Assume Σ is satisfiable. To proceed by contradiction, suppose that: $\Sigma \not\models \beta_1$ and $\Sigma \not\models \beta_2$. This implies there exist models ω_1 and ω_2 such that:

1. $\omega_1 \models \Sigma$ and $\omega_1 \not\models \beta_1$ (equivalently, $\omega_1 \models \neg\beta_1$).
2. $\omega_2 \models \Sigma$ and $\omega_2 \not\models \beta_2$ (equivalently, $\omega_2 \models \neg\beta_2$).

Since $\Sigma \equiv f(A) \wedge f(B)$, any model of Σ must satisfy both projected components:

$$\omega_1 \models f(A) \wedge f(B) \implies \omega_1[A] \models f(A)$$
$$\omega_2 \models f(A) \wedge f(B) \implies \omega_2[B] \models f(B)$$

Because $A \cap B = \emptyset$, we can construct a new assignment ω by combining the relevant parts of ω_1 and ω_2:

$$\omega = \omega_1[A] \cup \omega_2[B]$$

This assignment is well-defined and as we have

$$\omega[A] = \omega_1[A] \models f(A) \quad \text{and} \quad \omega[B] = \omega_2[B] \models f(B)$$

it follows that $\omega \models f(A) \wedge f(B)$, and therefore $\omega \models \Sigma$.

However, consider the clauses β_1 and β_2:

- Since $Var(\beta_1) \subseteq A$, the truth value of β_1 under ω depends only on assignments in A. As $\omega_1 \models \neg\beta_1$ and $\omega[A] = \omega_1[A]$, we have $\omega \models \neg\beta_1$.
- Similarly, since $Var(\beta_2) \subseteq B$, the truth value depends only on B. As $\omega_2 \models \neg\beta_2$ and $\omega[B] = \omega_2[B]$, we have $\omega \models \neg\beta_2$.

Consequently, we derive $\omega \models \neg\beta_1 \wedge \neg\beta_2$, which implies $\omega \models \neg(\beta_1 \vee \beta_2)$, or $\omega \models \neg\alpha$.

Contradiction: We have established that $\omega \models \Sigma$. Since $\alpha \in \Sigma$, every model of Σ must satisfy α. However, we showed that $\omega \models \neg\alpha$. □

Example 5 (Example 1 continued)**.** *Let us consider again the decision-DNNF depicted in Figure 1. Let us consider the formula $\Sigma_{|\neg x_1} = \{x_2 \vee x_3, x_3 \vee x_4 \vee x_5, \neg x_3 \vee x_4 \vee x_5\}$.*

This proposition implies that when a clause is shared between two sub-circuits linked by an AND node, it must functionally belong to one side or the other (or be tautological). Based on this, we define the *Local CNF* of a node N_i given a CNF formula Σ, denoted $cnf(N_i, \Sigma)$, as the subset of conditioned clauses projected onto the node's variables, retaining only those strictly entailed by the sub-circuit:

$$cnf(N_i, \Sigma) = \{\, \alpha \cap Var(N_i) \mid \alpha \in \Sigma|_{Lits(N_i)} \text{ and } circ(N_i) \models \alpha \cap Var(N_i) \,\}$$

Example 6 (Example 1 continued)**.** *Revisiting the decision-DNNF shown in Figure 1, consider the conditioned formula: $\Sigma|_{\neg x_1} = \{x_2 \vee x_3, x_3 \vee x_4 \vee x_5, \neg x_3 \vee x_4 \vee x_5\}$. The local CNF projections for the sub-circuits are: $cnf(N_5, \Sigma) = \{x_2 \vee x_3\}$ and $cnf(N_6, \Sigma) = \{x_4 \vee x_5\}$. This illustrates the projection property, as the full conditioned formula entails the local clauses (e.g., $\Sigma|_{\neg x_1} \models x_4 \vee x_5$).*

In fact, the following corollary states that the local CNF projection is logically equivalent to the sub-circuit itself.

Corollary 1. *Let Σ be a CNF formula represented by a decision-DNNF Ψ and N_i a node of Ψ, then $circ(N_i) \equiv cnf(N_i, \Sigma)$.*

Proof. By Proposition 1, we have that

$$\Sigma|_{Lits(N_i)} \equiv \bigwedge_{N_j \in \mathcal{K}(\pi)} circ(N_j) \equiv circ(N_i) \wedge \Phi_{rest}$$

where $circ(N_i)$ and Φ_{rest} do not share any variables. We verify that this corresponds to $cnf(N_i, \Sigma) \wedge cnf(\Phi_{rest}, \Sigma)$ by considering any clause $\alpha \in \Sigma|_{Lits(N_i)}$. If $Var(\alpha) \subseteq Var(N_i)$, then $\alpha \in cnf(N_i, \Sigma)$. If $Var(\alpha) \subseteq Var(\Phi_{rest})$, then $\alpha \in cnf(\Phi_{rest}, \Sigma)$. If $Var(\alpha)$ is shared between N_i and Φ_{rest}, then by Proposition 2, this clause can be split into two parts $\alpha = \beta_{N_i} \vee \beta_{\Phi_{rest}}$. If $circ(N_i) \models \beta_{N_i}$ then $\beta_{N_i} \in cnf(N_i, \Sigma)$, and if $\Phi_{rest} \models \beta_{\Phi_{rest}}$ then $\beta_{\Phi_{rest}} \in cnf(\Phi_{rest}, \Sigma)$. Since $\Sigma|_{Lits(N_i)} \equiv circ(N_i) \wedge \Phi_{rest} \equiv cnf(N_i, \Sigma) \wedge cnf(\Phi_{rest}, \Sigma)$ and $Var(N_i) = Var(cnf(N_i, \Sigma))$ and $Var(\Phi_{rest}) = Var(cnf(\Phi_{rest}, \Sigma))$, it follows that $circ(N_i) \equiv cnf(N_i, \Sigma)$. □

Algorithm 1: decision-DNNF Minimization

Input : A decision-DNNF Ψ, the original CNF Σ
Output: A minimized decision-DNNF Ψ

```
// 1. Semantic Hashing
```
1 Compute $H(u) = \langle Var(u), mc(u) \rangle \; \forall u \in \Psi$;
2 $Buckets \leftarrow$ Group nodes u by value of $H(u)$;
```
// 2. Verification and Merging
```
3 **foreach** *bucket* $B \in Buckets$ *such that* $|B| > 1$ **do**
4 **while** $B \neq \emptyset$ **do**
5 $u \leftarrow \text{Pop}(B)$; `// Select pivot`
6 **foreach** $v \in B$ **do**
 `// Check Mutual Entailment`
7 **if** $circ(u) \models cnf(v, \Sigma)$ ***and*** $circ(v) \models cnf(u, \Sigma)$ **then**
8 Merge(v, u);
9 $B \leftarrow B \setminus \{v\}$;
10 **return** Ψ;

Finally, to verify if two candidate nodes N_i and N_j (where $H(N_i) = H(N_j)$) are equivalent, it is possible to take advantage of the following proposition.

Proposition 3. *Let Σ be a CNF formula represented by a decision-DNNF Ψ. Let N_1 and N_2 be two nodes of Ψ. If $circ(N_1) \models cnf(N_2, \Sigma)$ and $circ(N_2) \models cnf(N_1, \Sigma)$, then $circ(N_1) \equiv circ(N_2)$.*

Proof. Corollary 1 implies that $circ(N_1) \equiv cnf(N_1, \Sigma)$ and, likewise, that $circ(N_2) \equiv cnf(N_2, \Sigma)$. Thus, if $circ(N_1) \models cnf(N_2, \Sigma)$ then $circ(N_1) \models circ(N_2)$, and if $circ(N_2) \models cnf(N_1, \Sigma)$ then $circ(N_2) \models circ(N_1)$. Consequently, $circ(N_1) \equiv circ(N_2)$. □

Because clause entailment can be decided in polynomial time for decision-DNNF [Darwiche and Marquis, 2002], computing this projection is tractable. Notably, this tractability extends to the broader class of DNNFs as well. Therefore, it is possible to verify if $circ(N_i) \models cnf(N_j, \Sigma)$ in polynomial time. By performing this check in both directions (mutual entailment), we can determine the equivalence of two sub-circuits efficiently.

Algorithm 1 provides a detailed summary of the proposed minimization framework. The procedure begins by computing the semantic hash $H(u)$ for every node in a bottom-up pass (line 1). Nodes sharing the same hash are grouped into buckets, representing potential equivalence classes (line 2). The algorithm then iteratively refines these buckets. For each bucket, it selects a representative pivot u and compares other candidates v against it (lines 6–10). To verify equivalence, it checks if the pivot sub-circuit $circ(u)$ entails the Local CNF of the candidate $cnf(v, \Sigma)$ (and vice-versa). If this mutual entailment holds (Line 7), the nodes are confirmed to be equivalent; consequently, v is merged into u (line 8), and v is removed from the bucket (line 9).

It is important to note that the decision-DNNF produced by Algorithm 1 is minimal in the sense that no two distinct sub-circuits represent equivalent Boolean functions.

Proposition 4 (Minimality)**.** *Let Ψ' be the decision-DNNF returned by Algorithm 1. For any two distinct nodes $N_i, N_j \in \Psi'$, $circ(N_i) \not\equiv circ(N_j)$.*

Proof. We proceed by contradiction. Assume there exist two distinct nodes N_i, N_j in Ψ' such that $circ(N_i) \equiv circ(N_j)$. Since the nodes are equivalent, they must share the same variable scope and model count (as established in Section 3.1). Thus, $H(N_i) = H(N_j)$, meaning both nodes were initially placed in the same bucket B during Phase 1.

Consider the iteration of the **While** loop (Line 6) where the first of these two nodes, say N_i, is selected as the pivot u. At this moment, since N_j is distinct and has not yet been processed as a pivot, it must be present in B (or have been merged into an equivalent node N_k, which would transitively imply $N_i \equiv N_k$). The algorithm then performs the mutual entailment check between $u = N_i$ and $v = N_j$ (Line 9). Since we assumed $circ(N_i) \equiv circ(N_j)$, by the previous Proposition, the conditions $circ(N_i) \models cnf(N_j, \Sigma)$ and $circ(N_j) \models cnf(N_i, \Sigma)$ are satisfied. Consequently, N_j is merged into N_i (Line 10) and removed from the bucket. Therefore, N_j cannot appear as a distinct node in the final output Ψ', which contradicts the assumption. □

4 Related Work

Complexity of Knowledge Compilation. Despite the inferential tractability of d-DNNF, the size of compiled circuits remains a critical bottleneck. Theoretical results have established unconditional lower bounds for knowledge compilation languages, proving that succinct representations for general propositional theories are infeasible under standard complexity assumptions [Bova *et al.*, 2016; Beame *et al.*, 2017].

Consequently, d-DNNF circuits often exhibit exponential growth relative to the input formula size. While heuristic optimizations exist [Koriche *et al.*, 2013], they typically target specific sub-classes or lack a systematic approach to post-compilation minimization. Our work operates within these theoretical bounds but seeks to minimize the *practical* representation size of tractable instances.

Circuit Minimization and FRAIGs. In the broader field of logic synthesis, AND-Inverter Graphs (AIGs) serve as a standard representation for boolean networks [Kuehlmann *et al.*, 2002]. To optimize these structures, the *Functionally Reduced AIG* (FRAIG) framework was developed, integrating structural hashing with SAT-based functional reduction [Mishchenko *et al.*, 2005].

The core mechanism of FRAIGs involves identifying potentially equivalent nodes via simulation and verifying their equivalence using a SAT solver (SAT sweeping). This approach is widely used in hardware verification and equivalence checking [Goldberg *et al.*, 2001; Biere, 2021].

Since d-DNNFs are essentially Boolean circuits, FRAIG-style reduction is theoretically applicable. However, the reliance on SAT sweeping introduces a severe computational bottleneck. Confirming the equivalence of cut-points requires repeated calls to an NP-complete oracle, which contradicts the goal of using compiled languages (like d-DNNF) specifically designed to avoid such complexity.

In contrast, our approach replaces the expensive SAT solver with a polynomial-time semantic hash based on model counting, leveraging the inherent properties of the d-DNNF itself to facilitate efficient equivalence checking.

Redundancy in Knowledge Compilation. Specific to the d-DNNF context, recent work has addressed size reduction by targeting artifacts introduced during compilation. Notably, compilers that rely on Tseitin transformations introduce auxiliary variables that may become redundant during downstream tasks. Kiesel et al. [Kiesel and Eiter, 2023] and Derkinderen [Derkinderen, 2024] demonstrated that when these variables are existentially quantified (projected out), the resulting circuits often contain large tautological substructures. Derkinderen proposed a bottom-up traversal to detect and prune these artifacts efficiently.

While effective, this technique is limited in scope: it primarily addresses redundancies arising from *forgetting* (existential quantification). It does not address general functional redundancies in standard compilation scenarios where variables are preserved. Our method generalizes this minimization objective, identifying functional equivalence across arbitrary sub-circuits regardless of their origin.

Equivalence of two decision-DNNFs. It is currently unknown whether the equivalence of two arbitrary decision-DNNFs can be decided in polynomial time; the question remains open [Darwiche and Marquis, 2002]. However, randomized algorithms allow for equivalence checking with high statistical confidence. This strategy is based on the arithmetization method originally proposed for FBDDs (a subclass of decision-DNNFs without decomposable AND nodes) in [Blum *et al.*, 1980]. The key insight is to transform the circuit into a multilinear polynomial over a finite field and evaluate it at a random assignment. If the computed values differ, the circuits are definitely not equivalent; if they match, they are equivalent with high probability.

Since the detection of non-equivalence is always certain, the non-equivalence problem belongs to the complexity class RP, placing the equivalence problem in co-RP. As shown in [Darwiche and Huang, 2002], this technique extends effectively to decision-DNNFs. In contrast to our work, which provides a deterministic check by leveraging the source CNF, this method is probabilistic. Furthermore, to the best of our knowledge, no minimization algorithm leveraging this randomized check has been developed to date.

5 Experimental Evaluation

Since there are no existing complete reduction algorithms specifically designed for decision-DNNFs, we evaluate our technique against **FRAIG**. We integrated our algorithm into the **decdnnf_rs** reasoner [Lagniez and Lonca, 2024; Lonca and Lagniez, 2026] and performed comparisons using the **FRAIG** implementation in **abc**, executed via the `&fraig -y` command.[1] Detailed logs and experimental artifacts are available at https://zenodo.org/records/20165767.

We compared both approaches using 3,828 benchmarks drawn from recent studies. First we considered 940 instances from enumeration studies [Spallitta *et al.*, 2024] divided into four sets: Binary Clauses (binary – 50 instances), random 3-SAT problems (rnd3sat – 410 instances), Random-3-SAT Instances with Controlled Backbone Size (CBS – 1000 instances), and Random-3-SAT Instances and Backbone-minimal Subinstances (BMS – 500 instances); Then we used 246 instances from three recent model counting competitions (https://mccompetition.org), 197 from [Lagniez and Marquis, 2017b] to evaluate **d4** and 1425 benchmarks from studies on uniform sampling [Sharma *et al.*, 2018; Lagniez and Lonca, 2025].

We employed **d4**[2] to compile the source CNF benchmarks into decision-DNNF representations. For our evaluation, we restricted the dataset to the 3,596 benchmarks where compilation completed successfully within a one-hour timeout and the resulting formula was satisfiable (i.e., admitted at least one model). Experiments were conducted on machines equipped with Intel® Xeon® E5–2637 v4 CPUs and 50 GB of RAM, running Rocky Linux 9.5.

To facilitate comparison with **FRAIG**, we developed an utility to translate decision-DNNFs into ASCII AIG format, which were subsequently converted to binary AIGs using the **aigtoaig** tool from the **AIGER** suite [Biere, 2007; Biere *et al.*, 2011]. While **FRAIG** leverages this optimized binary input, **decdnnf_rs** currently processes the circuit structure directly, as it does not yet support a native binary format.

Translating decision-DNNFs produced by **d4** into AIGs can significantly increase the number of conjunction nodes as, to ensure succinctness, **d4** employs an edge-labeled representation where literals are associated with edges rather than nodes. In this format, literals on an edge represent assignments that must be propagated during path traversal.

This compact encoding allows, for example, a term of arbitrary length to be represented by just two nodes and a single labeled edge. Conversely, the AIG format is restricted to binary conjunctions, requiring $n-1$ nodes to represent a term of n literals. While this translation introduces nodes not present in the original decision-DNNF, many are redundant and can be merged via structural hashing when the instance is loaded by **abc**, prior to the application of **FRAIG**.

The reduction in node count through initial structural hashing is considered negligible in our analysis. This is justified by two factors: (i) the process primarily merges nodes that are artifacts of the AIG translation and were not present in the original decision-DNNF structure; and (ii) computational overhead of structural hashing (including I/O) is small, i.e., less than one second per average, peaking at 57 seconds, thus a negligible fraction of the 3,600 seconds time budget. Both **decdnnf_rs** and **FRAIG** were executed under the same timeout and resource constraints as the initial compilation phase.

We first analyze the CPU time required by each tool to perform circuit reduction, with the results summarized in Fig. 2. The scatter plot reveals that **decdnnf_rs** achieves significantly lower computation times than **FRAIG**, a result that aligns with the tools' respective theoretical complexities.

Notably, **decdnnf_rs** successfully processed a higher number of instances (3,540) within the timeout compared to

[1] https://github.com/berkeley-abc/abc, commit c18b9a2

[2] https://www.cril.univ-artois.fr/software/d4/

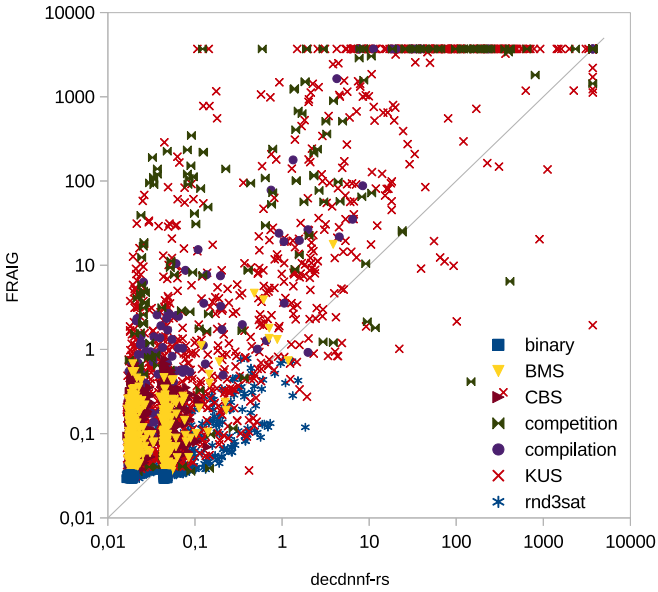

Figure 2: Comparison of CPU times (in seconds) used to reduce an instance. Each point represents an instance and gives the CPU time spent to compute the reduction with **decdnnf_rs** on the x-axis and **FRAIG** on the y-axis (lower is better). Dots on the top and right sides on the plot are instances that reached timeouts.

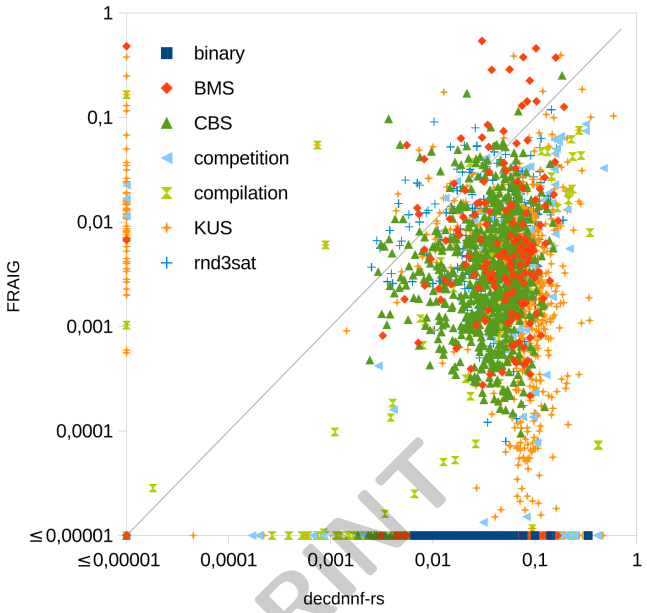

Figure 3: Comparison of the reduction achieved by **decdnnf_rs** and **FRAIG** respectively. Each point represents an instance and gives the reduction ratio between the number of edges after and before the reduction with **decdnnf_rs** on the x-axis and **FRAIG** on the y-axis (higher is better). Timeouts count as no reduction (points on axis).

FRAIG (3,341). Furthermore, on the subset of benchmarks completed by both approaches, **decdnnf_rs** required an average of only 3.06 s, whereas **FRAIG** averaged 29.23 s.

Next, we evaluate the reduction efficiency of **decdnnf_rs** compared to **FRAIG**. Figure 3 illustrates these results by plotting the ratio of eliminated edges for each instance. For our approach, the reduction is measured directly by comparing the edge count before and after the process. For **FRAIG**, we utilize the `ps` command in **abc** to retrieve the number of AIG nodes, noting that the edge count is exactly twice the node count. In Fig. 3, instances where a timeout occurred for **decdnnf_rs** or **FRAIG** are plotted along the leftmost or bottommost axes, respectively; this signifies that no reduction was successfully computed by that tool, while the competing approach may have completed.

The reduction analysis confirms the superior performance of the **decdnnf_rs** tool over **FRAIG**. On average, **decdnnf_rs** achieves a reduction of 5.00%, whereas **FRAIG** yields only 0.68%. Interestingly, all instance families benefit from the reduction provided by **decdnnf_rs**, with average gains ranging from 2.87% for binary instances to 6.65% for competition instances. In contrast, **FRAIG** fails to find any reduction for the binary family, and its highest average reduction (0.91%) is observed for the rnd3sat instances.

In theory, translating the reduced AIGs back into **decdnnf_rs** representations would allow for a more direct comparison of reduction efficiency. However, this approach is impractical for two primary reasons: (i) there is no guarantee that the circuit produced by the **FRAIG** command preserves the determinism property required for a valid **decdnnf_rs**; and (ii) it would be necessary to ensure that all auxiliary conjunction nodes introduced during the initial AIG translation are eliminated, rather than remaining as structural artifacts that would artificially inflate the final edge count.

6 Conclusion and Perspectives

In this paper, we proposed a novel framework for minimizing decision-DNNFs. Our approach leverages a dedicated semantic hashing technique to significantly prune the search space of candidate node pairs. Crucially, by exploiting the relationship between a circuit and its source CNF formula, we introduced a verification method based on *local CNF projection*, which enables polynomial-time equivalence checking.

We formally proved the resulting circuits to be minimal, in the sense that no two distinct sub-circuits represent the same Boolean function. Our experimental evaluation demonstrates that our minimization process is highly efficient, achieving up to a 10% reduction in edge count on standard benchmarks.

For future work, we plan to investigate whether this minimization framework can be extended to unconstrained d-DNNFs. We also aim to integrate this procedure directly into the compilation pipeline; by applying minimization dynamically (potentially via a dedicated parallel thread) we can significantly reduce the memory footprint of intermediate structures and mitigate memory exhaustion on challenging instances. Furthermore, we intend to incorporate probabilistic equivalence tests as an efficient early-rejection filter to quickly prune non-equivalent candidates before performing the full CNF-based verification. Finally, we will further exploit parallelism within the reduction algorithm itself by processing node buckets and equivalence checks concurrently to enhance overall scalability.

Acknowledgements

This work was partly supported by the ANR CERADOC project (ANR-25-CE23-3078). We also thank the anonymous reviewers for their insightful feedback and constructive suggestions.

References

[Amarù *et al.*, 2020] Luca G. Amarù, Felipe S. Marranghello, Eleonora Testa, Christopher Casares, Vinicius N. Possani, Jiong Luo, Patrick Vuillod, Alan Mishchenko, and Giovanni De Micheli. SAT-sweeping enhanced for logic synthesis. In *57th ACM/IEEE Design Automation Conference, DAC 2020, San Francisco, CA, USA, July 20-24, 2020*, pages 1–6. IEEE, 2020.

[Astesana *et al.*, 2010] Jean-Marc Astesana, Laurent Cosserat, and Hélène Fargier. Constraint-based vehicle configuration: A case study. In *22nd IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2010, Arras, France, 27-29 October 2010 - Volume 1*, pages 68–75. IEEE Computer Society, 2010.

[Beame *et al.*, 2017] Paul Beame, Jerry Li, Sudeepa Roy, and Dan Suciu. Exact model counting of query expressions: Limitations of propositional methods. *ACM Trans. Database Syst.*, 42(1):1:1–1:46, 2017.

[Biere *et al.*, 2011] Armin Biere, Keijo Heljanko, and Siert Wieringa. AIGER 1.9 and beyond. Technical Report 11/2, Institute for Formal Models and Verification, Johannes Kepler University, 4041 Linz, Austria, 2011.

[Biere *et al.*, 2021] Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors. *Handbook of Satisfiability - Second Edition*, volume 336 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 2021.

[Biere *et al.*, 2024a] Armin Biere, Katalin Fazekas, Mathias Fleury, and Nils Froleyks. Clausal congruence closure. In *27th International Conference on Theory and Applications of Satisfiability Testing, SAT 2024, Pune, India, August 21-24, 2024*, volume 305 of *LIPIcs*, pages 6:1–6:25. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024.

[Biere *et al.*, 2024b] Armin Biere, Katalin Fazekas, Mathias Fleury, and Nils Froleyks. Clausal equivalence sweeping. In *Formal Methods in Computer-Aided Design, FMCAD 2024, Prague, Czech Republic, October 15-18, 2024*, pages 1–6. IEEE, 2024.

[Biere, 2007] Armin Biere. The AIGER And-Inverter Graph (AIG) format version 20071012. Technical Report 07/1, Institute for Formal Models and Verification, Johannes Kepler University, Linz, Austria, 2007.

[Biere, 2021] Armin Biere. Bounded model checking. In *Handbook of Satisfiability - Second Edition*, volume 336 of *Frontiers in Artificial Intelligence and Applications*, pages 739–764. IOS Press, 2021.

[Blum *et al.*, 1980] Manuel Blum, Ashok K. Chandra, and Mark N. Wegman. Equivalence of free boolean graphs can be decided probabilistically in polynomial time. *Inf. Process. Lett.*, 10(2):80–82, 1980.

[Bova *et al.*, 2016] Simone Bova, Florent Capelli, Stefan Mengel, and Friedrich Slivovsky. Knowledge compilation meets communication complexity. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 1008–1014. IJCAI/AAAI Press, 2016.

[Brachman and Levesque, 2004] Ronald J. Brachman and Hector J. Levesque. *Knowledge Representation and Reasoning*. Elsevier, 2004.

[Cadoli and Donini, 1997] Marco Cadoli and Francesco M. Donini. A survey on knowledge compilation. *AI Commun.*, 10(3-4):137–150, 1997.

[Cook, 1971] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing, May 3-5, 1971, Shaker Heights, Ohio, USA*, pages 151–158. ACM, 1971.

[Darwiche and Huang, 2002] Adnan Darwiche and Jinbo Huang. Testing equivalence probabilistically. Technical Report D-130, UCLA, 2002.

[Darwiche and Marquis, 2002] Adnan Darwiche and Pierre Marquis. A knowledge compilation map. *J. Artif. Intell. Res.*, 17:229–264, 2002.

[Darwiche, 2002] Adnan Darwiche. A compiler for deterministic, decomposable negation normal form. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence and Fourteenth Conference on Innovative Applications of Artificial Intelligence, July 28 - August 1, 2002, Edmonton, Alberta, Canada*, pages 627–634. AAAI Press / The MIT Press, 2002.

[Darwiche, 2004] Adnan Darwiche. New advances in compiling CNF into decomposable negation normal form. In *Proceedings of the 16th Eureopean Conference on Artificial Intelligence, ECAI'2004, including Prestigious Applicants of Intelligent Systems, PAIS 2004, Valencia, Spain, August 22-27, 2004*, pages 328–332. IOS Press, 2004.

[Darwiche, 2023] Adnan Darwiche. Logic for explainable AI. In *38th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2023, Boston, MA, USA, June 26-29, 2023*, pages 1–11. IEEE, 2023.

[Derkinderen, 2024] Vincent Derkinderen. Pruning boolean d-DNNF circuits through tseitin-awareness. In *36th IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2024, Herndon, VA, USA, October 28-30, 2024*, pages 663–670. IEEE, 2024.

[Fargier and Marquis, 2006] Hélène Fargier and Pierre Marquis. On the use of partially ordered decision graphs in knowledge compilation and quantified boolean formulae. In *Proceedings, The Twenty-First National Conference on Artificial Intelligence and the Eighteenth Innovative Applications of Artificial Intelligence Conference, July 16-20, 2006, Boston, Massachusetts, USA*, pages 42–47. AAAI Press, 2006.

[Goldberg *et al.*, 2001] Evguenii I. Goldberg, Mukul R. Prasad, and Robert K. Brayton. Using SAT for combinational equivalence checking. In *Proceedings of the Conference on Design, Automation and Test in Europe, DATE*

2001, Munich, Germany, March 12-16, 2001, pages 114–121. IEEE Computer Society, 2001.

[Huang and Darwiche, 2005] Jinbo Huang and Adnan Darwiche. DPLL with a trace: From SAT to knowledge compilation. In *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK, July 30 - August 5, 2005*, pages 156–162. Professional Book Center, 2005.

[Kiesel and Eiter, 2023] Rafael Kiesel and Thomas Eiter. Knowledge compilation and more with SharpSAT-TD. In *Proceedings of the 20th International Conference on Principles of Knowledge Representation and Reasoning, KR 2023, Rhodes, Greece, September 2-8, 2023*, pages 406–416, 2023.

[Koriche *et al.*, 2013] Frédéric Koriche, Jean-Marie Lagniez, Pierre Marquis, and Samuel Thomas. Knowledge compilation for model counting: Affine decision trees. In *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013*, pages 947–953. IJCAI/AAAI, 2013.

[Kuehlmann *et al.*, 2002] Andreas Kuehlmann, Viresh Paruthi, Florian Krohm, and Malay K. Ganai. Robust boolean reasoning for equivalence checking and functional property verification. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, 21(12):1377–1394, 2002.

[Kuehlmann, 2004] Andreas Kuehlmann. Dynamic transition relation simplification for bounded property checking. In *2004 International Conference on Computer-Aided Design, ICCAD 2004, San Jose, CA, USA, November 7-11, 2004*, pages 50–57. IEEE Computer Society / ACM, 2004.

[Lagniez and Lonca, 2024] Jean-Marie Lagniez and Emmanuel Lonca. Leveraging decision-DNNF compilation for enumerating disjoint partial models. In *Proceedings of the 21st International Conference on Principles of Knowledge Representation and Reasoning, KR 2024, Hanoi, Vietnam. November 2-8, 2024*, 2024.

[Lagniez and Lonca, 2025] Jean-Marie Lagniez and Emmanuel Lonca. Enhancing query efficiency for D-DNNF representations through preprocessing. In *Logics in Artificial Intelligence - 19th European Conference, JELIA 2025, Kutaisi, Georgia, September 1-4*, volume 16094 of *Lecture Notes in Computer Science*, pages 125–140. Springer, 2025.

[Lagniez and Marquis, 2017a] Jean-Marie Lagniez and Pierre Marquis. An improved decision-DNNF compiler. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 667–673. ijcai.org, 2017.

[Lagniez and Marquis, 2017b] Jean-Marie Lagniez and Pierre Marquis. An improved decision-DNNF compiler. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 667–673. ijcai.org, 2017.

[Lonca and Lagniez, 2026] Emmanuel Lonca and Jean-Marie Lagniez. Dynamic blocked clause elimination for projected model counting. In *29th International Conference on Theory and Applications of Satisfiability Testing, SAT 2026, Lisbon, Portugal, July 20-23, 2026*, LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2026.

[Lu *et al.*, 2003] Feng Lu, Li-C. Wang, Kwang-Ting Cheng, and Ric C.-Y. Huang. A circuit SAT solver with signal correlation guided learning. In *2003 Design, Automation and Test in Europe Conference and Exposition (DATE 2003), 3-7 March 2003, Munich, Germany*, pages 10892–10897. IEEE Computer Society, 2003.

[Mishchenko *et al.*, 2005] Alan Mishchenko, Satrajit Chatterjee, Roland Jiang, and Robert K. Brayton. Fraigs: A unifying representation for logic synthesis and verification. Technical report, Department of EECS, University of California, Berkeley, 2005.

[Muise *et al.*, 2012] Christian J. Muise, Sheila A. McIlraith, J. Christopher Beck, and Eric I. Hsu. Dsharp: Fast d-DNNF compilation with sharpSAT. In *Advances in Artificial Intelligence - 25th Canadian Conference on Artificial Intelligence, Canadian AI 2012, Toronto, ON, Canada, May 28-30, 2012. Proceedings*, volume 7310 of *Lecture Notes in Computer Science*, pages 356–361. Springer, 2012.

[Plaisted, 2015] David A. Plaisted. History and prospects for first-order automated deduction. In *Automated Deduction - CADE-25 - 25th International Conference on Automated Deduction, Berlin, Germany, August 1-7, 2015, Proceedings*, volume 9195 of *Lecture Notes in Computer Science*, pages 3–28. Springer, 2015.

[Rungta, 2022] Neha Rungta. A billion SMT queries a day (invited paper). In *Computer Aided Verification - 34th International Conference, CAV 2022, Haifa, Israel, August 7-10, 2022, Proceedings, Part I*, volume 13371 of *Lecture Notes in Computer Science*, pages 3–18. Springer, 2022.

[Sharma *et al.*, 2018] Shubham Sharma, Rahul Gupta, Subhajit Roy, and Kuldeep S. Meel. Knowledge compilation meets uniform sampling. In *LPAR-22. 22nd International Conference on Logic for Programming, Artificial Intelligence and Reasoning, Awassa, Ethiopia, 16-21 November 2018*, volume 57 of *EPiC Series in Computing*, pages 620–636. EasyChair, 2018.

[Spallitta *et al.*, 2024] Giuseppe Spallitta, Roberto Sebastiani, and Armin Biere. Disjoint partial enumeration without blocking clauses. In *Thirty-Eighth AAAI Conference on Artificial Intelligence, Vancouver, Canada*, pages 8126–8135. AAAI Press, 2024.