On CNF Conversion for SAT Enumeration

Gabriele Masina
DISI, University of Trento, Italy

Giuseppe Spallitta
DISI, University of Trento, Italy

Roberto Sebastiani
DISI, University of Trento, Italy

Abstract

Modern SAT solvers are designed to handle problems expressed in Conjunctive Normal Form (CNF) so that non-CNF problems must be CNF-ized upfront, typically by using variants of either Tseitin or Plaisted and Greenbaum transformations. When passing from solving to enumeration, however, the capability of producing partial satisfying assignments that are as small as possible becomes crucial, which raises the question of whether such CNF encodings are also effective for enumeration.

In this paper, we investigate both theoretically and empirically the effectiveness of CNF conversions for disjoint SAT enumeration. On the negative side, we show that: (i) Tseitin transformation prevents the solver from producing short partial assignments, thus seriously affecting the effectiveness of enumeration; (ii) Plaisted and Greenbaum transformation overcomes this problem only in part. On the positive side, we show that combining Plaisted and Greenbaum transformation with NNF preprocessing upfront —which is typically not used in solving— can fully overcome the problem and can drastically reduce both the number of partial assignments and the execution time.

2012 ACM Subject Classification Theory of computation → Automated reasoning; Hardware → Theorem proving and SAT solving; Computing methodologies → Representation of Boolean functions

Keywords and phrases CNF conversion, AllSAT, AllSMT

Supplementary Material *Software (Source Code)*: `https://github.com/masinag/allsat-cnf`

Funding We acknowledge the support of the MUR PNRR project FAIR – Future AI Research (PE00000013), under the NRRP MUR program funded by the NextGenerationEU. The work was partially supported by the project "AI@TN" funded by the Autonomous Province of Trento.

1 Introduction

State-of-the-art SAT and SMT solvers deal very efficiently with formulas expressed in Conjunctive Normal Form (CNF). In real-world scenarios, however, it is common for problems to be expressed as non-CNF formulas. Hence, these problems must be converted into CNF before being processed by the solver. This conversion is generally done by using variants of the Tseitin [20] or the Plaisted and Greenbaum [16] transformations, which generate a linear-size equisatisfiable CNF formula by labelling sub-formulas with fresh Boolean atoms. These transformations can be employed also for SAT and SMT enumeration (also referred to in the literature as AllSAT and AllSMT), by projecting the models on the original atoms only.

When passing from SAT to AllSAT, however, the capability of enumerating partial satisfying assignments that are as small as possible is crucial, because each prevents from enumerating a number of total assignments that is exponential w.r.t. the number of unassigned atoms. This raises the question of whether CNF encodings conceived for solving are also effective for enumeration. To the best of our knowledge, however, no research has yet been published to analyse how the different CNF encodings may affect the effectiveness of the solvers for AllSAT and AllSMT.

In this paper, we investigate, both theoretically and empirically, the effectiveness of CNF conversion for enumeration. We focus on AllSAT, restricting to disjoint enumeration. We expect analogous results for AllSMT. The contribution of this paper is twofold. First, on the negative side, we show that the commonly employed CNF transformations for SAT are not suitable for AllSAT. In particular, we notice that the Tseitin encoding introduces top-level label definitions for sub-formulas with double implications, which need to be satisfied as well and thus prevent the solver from producing short partial assignments. We also notice that the Plaisted and Greenbaum transformation solves this problem only in part by labelling sub-formulas only with single implications if they occur with single polarity, but it has similar issues to the Tseitin transformation when sub-formulas occur with both polarities. Second, on the positive side, we show that converting the formula into Negation Normal Form (NNF) before applying the Plaisted and Greenbaum transformation can fix the problem and drastically improve the effectiveness of the enumeration process by up to orders of magnitude.

This analysis is confirmed by an experimental evaluation of non-CNF problems originating from both synthetic and real-world-inspired applications. The results confirm the theoretical analysis, showing that the combination of NNF with the Plaisted and Greenbaum CNF allows for a significant reduction in both the number of partial assignments and the execution time.

Related Work

The impact of using different CNF encodings on the performance for SAT and SMT solving has been widely studied in the literature [3, 10, 2, 11].

Beyond the basic task of SAT and SMT solving, several applications in probabilistic reasoning require quantifying the number of solutions of a SAT or SMT formula. Whereas for some applications it is sufficient to count the number of satisfying assignments, others require to enumerate all of them. In particular, SAT and SMT disjoint enumeration play a foundational role in probabilistic reasoning frameworks such as #SMT [5] and Weighted Model Integration [13, 14, 18]. Specifically, in the case of #SMT($\mathcal{LRA}$) we need to sum up the volumes corresponding to each of the models, whereas in WMI we need to integrate some function w over the polytopes defined by each of the models. Hence, in these cases, it is essential to enumerate disjoint partial models that are as small and as few as possible.

The problem of model minimization for Tseitin-encoded problems was addressed by [9]. They first propose to simplify the formula by considering its original structure and the current model; then they use iterative calls to a SAT solver to obtain a minimal model by imposing increasingly tighter cardinality constraints. This approach can be used to find a single short model, but it can be very expensive and thus it is unsuitable for model enumeration.

Content

The paper is organized as follows. In §2 we introduce the theoretical background necessary to understand the rest of the paper. In §3 we analyse the problem of the classical CNF-izations when used for AllSAT. In §4 we propose one possible solution, whose effectiveness is evaluated on both synthetic and real-world inspired benchmarks in §5. We conclude the paper in §6, drawing some final remarks and indicating possible future work.

2 Background

This section introduces the notation and the theoretical background necessary to understand what is presented in this paper. We recall the standard syntax, semantics, and results of propositional logic, and the fundamental ideas behind SAT enumeration and projected enumeration implemented by modern AllSAT solvers.

2.1 Propositional Logic

In this section, we summarize some basic definitions and results of propositional logic.

Notation and Terminology

In the paper, we adopt the following conventions. We refer to propositional atoms with capital letters, such as A and B. Propositional formulas are referred to with Greek letters such as φ, ψ. Total truth assignments are denoted by η, while partial truth assignments are denoted by μ. The symbols $\mathbf{A} \stackrel{\text{def}}{=} \{A_1, \ldots, A_N\}$ and $\mathbf{B} \stackrel{\text{def}}{=} \{B_1, \ldots, B_K\}$ denote disjoint sets of propositional atoms. We denote Boolean constants by $\mathcal{B} \stackrel{\text{def}}{=} \{\top, \bot\}$.

A *propositional formula* φ can be defined recursively as follows. The Boolean constants $\top$ and $\bot$ are formulas; a Boolean atom A and its negation $\neg A$ are formulas, also referred to as *literals*; a connection of two formulas φ and ψ by one of the connectors $\wedge, \vee, \rightarrow, \leftrightarrow$ is a formula. A sub-formula occurs with *positive* [resp. *negative*] *polarity* (also *positively* [resp. *negatively*]) if it occurs under an even [resp. odd] number of nested negations. Specifically, φ occurs positively in φ; if $\neg\varphi_1$ occurs positively [resp. negatively] in φ, then φ_1 occurs negatively [resp. positively] in φ; if $\varphi_1 \wedge \varphi_2$ or $\varphi_1 \vee \varphi_2$ occur positively [resp. negatively] in φ, then φ_1 and φ_2 occur positively [resp. negatively] in φ; if $\varphi_1 \rightarrow \varphi_2$ occurs positively [resp. negatively] in φ, then φ_1 occurs negatively [resp. positively] and φ_2 occurs positively [resp. negatively] in φ; if $\varphi_1 \leftrightarrow \varphi_2$ occurs in φ, then φ_1 and φ_2 occur both positively and negatively in φ.

Negation Normal Form

A Boolean formula is in *Negation Normal Form (NNF)* iff it is given only by the recursive applications of $\wedge$ and $\vee$ to literals. An NNF formula can be conveniently represented as a Directed Acyclic Graph (DAG) —that is, as a single-root and/or DAG with literals as leaves. We call the *size* of an NNF DAG the sum of the numbers of its nodes and arcs. A formula can be converted into NNF by recursively rewriting implications $(\alpha \rightarrow \beta)$ as $(\neg\alpha \vee \beta)$ and equivalences $(\alpha \leftrightarrow \beta)$ as $(\neg\alpha \vee \beta) \wedge (\alpha \vee \neg\beta)$, and then by recursively "pushing down" the negations: $\neg(\alpha \vee \beta)$ as $(\neg\alpha \wedge \neg\beta)$, $\neg(\alpha \wedge \beta)$ as $(\neg\alpha \vee \neg\beta)$ and $\neg\neg\alpha$ as α. We have the following fact.

▶ **Property 1.** *Let φ be a Boolean formula and $\mathsf{NNF}(\varphi)$ be the NNF formula resulting from converting φ into NNF as described above. If $\mathsf{NNF}(\varphi)$ is represented as a DAG, then its size is linear w.r.t. the original one.*

(Although this fact is well-known, we provide a formal proof in Appendix A.) Intuitively, we only need at most 2 nodes for each sub-formula φ_i of φ, representing $\mathsf{NNF}(\varphi_i)$ and $\mathsf{NNF}(\neg\varphi_i)$ for positive and negative occurrences of φ_i respectively. These nodes are shared among up to exponentially-many branches generated by expanding the nested iffs.

CNF Transformations

A Boolean formula is in *Conjunctive Normal Form (CNF)* iff it is a conjunction ($\wedge$) of clauses, where a clause is a disjunction ($\vee$) of literals. Numerous CNF transformation procedures, commonly referred to as CNF-izations, have been proposed in the literature. In the next paragraph, we summarize the three most frequently employed techniques.

The *Classic CNF-ization* ($\mathsf{CNF_{DM}}$) converts any propositional formula into a logically equivalent formula in CNF by applying DeMorgan's rules. First, it converts the formula into NNF. Second, it recursively rewrites sub-formulas $\alpha \vee (\beta \wedge \gamma)$ as $(\alpha \vee \beta) \wedge (\alpha \vee \gamma)$ to distribute $\vee$ over $\wedge$, until the formula is in CNF. The principal limitation of this transformation lies in the possible exponential growth of the resulting formula compared to the original (e.g. when the formula is in DNF), making it unsuitable for modern SAT solvers [1].

The *Tseitin CNF-ization* ($\mathsf{CNF_{Ts}}$) [20] avoids this exponential blow-up by labelling each sub-formula φ_i with a fresh Boolean atom B_i, which is used as a placeholder for the sub-formula. Specifically, it consists in applying recursively bottom-up the rewriting rule $\varphi \Longrightarrow \varphi[\varphi_i|B_i] \wedge \mathsf{CNF_{DM}}(B_i \leftrightarrow \varphi_i)$ until the resulting formula is in CNF, where $\varphi[\varphi_i|B_i]$ is the formula obtained by substituting in φ every occurrence of φ_i with B_i.

The *Plaisted and Greenbaum CNF-ization* ($\mathsf{CNF_{PG}}$) [16] is a variant of the $\mathsf{CNF_{Ts}}$ that exploits the polarity of sub-formulas to reduce the number of clauses of the final formula. Specifically, if a sub-formula φ_i appears only with positive [resp. negative] polarity, then it can be labelled with a single implication as $\mathsf{CNF_{DM}}(B_i \rightarrow \varphi_i)$ [resp. $\mathsf{CNF_{DM}}(B_i \leftarrow \varphi_i)$].

With both $\mathsf{CNF_{Ts}}$ and $\mathsf{CNF_{PG}}$, due to the introduction of the label variables, the final formula does not preserve the equivalence with the original formula but only the equisatisfiability. Moreover, they also have a stronger property. If $\varphi(\mathbf{A})$ is a non-CNF formula and $\psi(\mathbf{A} \cup \mathbf{B})$ is either the $\mathsf{CNF_{Ts}}$ or the $\mathsf{CNF_{PG}}$ encoding of φ, where $\mathbf{B}$ are the fresh Boolean atoms introduced by the transformation, then $\varphi(\mathbf{A}) \equiv \exists \mathbf{B}.\psi(\mathbf{A} \cup \mathbf{B})$.

Total and partial truth assignments

Given a set of Boolean atoms $\mathbf{A}$, a *total truth assignment* is a total map $\eta : \mathbf{A} \longmapsto \mathcal{B}$. A *partial truth assignment* is a partial map $\mu : \mathbf{A} \longmapsto \mathcal{B}$. Notice that a partial truth assignment represents 2^K total truth assignments, where K is the number of unassigned variables by μ. With a little abuse of notation, we sometimes represent a truth assignment either as a set, s.t. $\mu \stackrel{\text{def}}{=} \{A \mid \mu(A) = \top\} \cup \{\neg A \mid \mu(A) = \bot\}$, or as a cube, s.t. $\mu \stackrel{\text{def}}{=} \bigwedge_{\mu(A)=\top} A \wedge \bigwedge_{\mu(A)=\bot} \neg A$. If $\mu_1 \subseteq \mu_2$ [resp. $\mu_1 \subset \mu_2$] we say that μ_1 is a *sub-assignment* [resp. *strict sub-assignment*] of μ_2 and that μ_2 is a *super-assignment* [resp. *strict super-assignment*] of μ_1. We denote with $\varphi|_\mu$ the *residual of* φ *under* μ, i.e. the formula obtained by substituting in φ each $A_i \in \mathbf{A}$ with $\mu(A_i)$, and by recursively applying the standard propagation rules of truth values through Boolean operators.

Given a set of Boolean atoms $\mathbf{A}$ and a formula $\varphi(\mathbf{A})$, we say that a *[partial or total] truth assignment* $\mu : \mathbf{A} \longmapsto \mathcal{B}$ *satisfies* φ, denoted as $\mu \models \varphi$, iff $\varphi|_\mu = \top$[1]. If $\mu \models \varphi$, then we say that μ is a *model* of φ. A partial truth assignment μ is *minimal* for φ iff $\mu \models \varphi$ and every strict sub-assignment $\mu' \subset \mu$ is such that $\mu' \not\models \varphi$.

Most of the modern SAT and SMT solvers do not deal directly with non-CNF formulas,

[1] The definition of satisfiability by partial assignment may present some ambiguities for non-CNF and existentially-quantified formulas [17, 15]. Here we adopt the above definition because it is the easiest to implement, and it is the one typically used by state-of-the-art SAT solvers. We refer to [17, 15] for details.

rather they convert them into CNF by using either $\mathsf{CNF_{Ts}}$ or $\mathsf{CNF_{PG}}$. As seen in the previous paragraph, since these transformations introduce fresh atoms into the resulting formulas, a model of $\varphi(\mathbf{A})$ can be found as a truth assignment satisfying $\exists\mathbf{B}.\psi(\mathbf{A}\cup\mathbf{B})$. Given two disjoint sets of Boolean atoms $\mathbf{A},\mathbf{B}$ and a CNF formula $\psi(\mathbf{A}\cup\mathbf{B})$, we say that a *[partial or total] truth assignment* $\mu^{\mathbf{A}}:\mathbf{A}\longmapsto\mathcal{B}$ *satisfies* $\exists\mathbf{B}.\psi$ iff there exists a total truth assignment $\eta^{\mathbf{B}}:\mathbf{B}\longmapsto\mathcal{B}$ such that $\mu^{\mathbf{A}}\cup\eta^{\mathbf{B}}:\mathbf{A}\cup\mathbf{B}\longmapsto\mathcal{B}$ satisfies ψ.

We have the following fact, for which we provide a complete proof in Appendix B.

▶ **Property 2.** *Consider a Boolean formula* $\varphi(\mathbf{A})$*, and let* $\mathsf{NNF}(\varphi)$ *be its NNF DAG. Consider a partial assignment* $\mu^{\mathbf{A}}$ *on* $\mathbf{A}$*. Then* $\varphi|_{\mu^{\mathbf{A}}}=v$ *iff* $\mathsf{NNF}(\varphi)|_{\mu^{\mathbf{A}}}=v$*, for* $v\in\{\top,\bot\}$*.*

2.2 AllSAT and Projected AllSAT

AllSAT is the task of enumerating all the models of a propositional formula. In this paper, we focus on the enumeration of disjoint models, that is, pairwise mutually-inconsistent models. Given a Boolean formula φ, we denote with $\mathcal{TTA}(\varphi)\stackrel{\text{def}}{=}\{\eta_1,\ldots,\eta_j\ldots\eta_M\}$ the set of all total truth assignments satisfying φ. We denote with $\mathcal{TA}(\varphi)\stackrel{\text{def}}{=}\{\mu_1,\ldots,\mu_i\ldots,\mu_N\}$ a set of partial truth assignments satisfying φ s.t.:

(a) every $\eta\in\mathcal{TTA}(\varphi)$ is a super-assignment of some $\mu\in\mathcal{TA}(\varphi)$;

(b) every pair $\mu_i,\mu_j\in\mathcal{TA}(\varphi)$ assigns opposite truth value to at least one atom.

Notice that, whereas $\mathcal{TTA}(\varphi)$ is unique, multiple $\mathcal{TA}(\varphi)$s are admissible for the same formula φ, including $\mathcal{TTA}(\varphi)$. AllSAT is the task of enumerating either $\mathcal{TTA}(\varphi)$ or a set $\mathcal{TA}(\varphi)$. Typically, AllSAT solvers aim at enumerating a set $\mathcal{TA}(\varphi)$ as small as possible, since every partial model prevents from enumerating a number of total models that is exponential w.r.t. the number of unassigned atoms, so that to save computational space and time.

The enumeration of a $\mathcal{TA}(\varphi)$ for a non-CNF formula φ is typically implemented by first converting it into CNF, and then enumerating its models by means of *Projected AllSAT*. Specifically, let $\varphi(\mathbf{A})$ be a non-CNF formula and let $\psi(\mathbf{A}\cup\mathbf{B})$ be the result of applying either $\mathsf{CNF_{Ts}}$ or $\mathsf{CNF_{PG}}$ to φ, where $\mathbf{B}$ is the set of Boolean atoms introduced by either transformation. $\mathcal{TA}(\varphi)$ is enumerated via Projected AllSAT as $\mathcal{TA}(\exists\mathbf{B}.\psi)$, i.e. as a set of (partial) truth assignments over $\mathbf{A}$ that can be extended to total models of ψ over $\mathbf{A}\cup\mathbf{B}$. We refer to the general schema described in [12], which we briefly recap here.

Let $\psi(\mathbf{A}\cup\mathbf{B})$ be a CNF formula over two disjoint sets of Boolean variables $\mathbf{A},\mathbf{B}$, where $\mathbf{A}$ is a set of *relevant atoms* s.t. we want to enumerate a $\mathcal{TA}(\exists\mathbf{B}.\psi)$. The solver enumerates one-by-one partial truth assignments $\mu_1,\ldots,\mu_i,\ldots\mu_N$, where each $\mu_i\stackrel{\text{def}}{=}\mu_i^{\mathbf{A}}\cup\eta_i^{\mathbf{B}}$ is s.t.:

(i) (*satisfiability*) $\mu_i\models\psi$;

(ii) (*disjointness*) for each $j<i$, $\mu_i^{\mathbf{A}},\mu_j^{\mathbf{A}}$ assign opposite truth values to some atom in $\mathbf{A}$;

(iii) (*minimality*) $\mu_i^{\mathbf{A}}$ is *minimal*, meaning that no literal can be dropped from it without losing properties (i) and (ii).

A basic disjoint AllSAT procedure (implemented e.g. in MathSAT [6]) works as follows. At each step i, it finds a total truth assignment $\eta_i\stackrel{\text{def}}{=}\eta_i^{\mathbf{A}}\cup\eta_i^{\mathbf{B}}$ s.t. $\eta_i\models\psi_i$, where $\psi_i\stackrel{\text{def}}{=}\psi\wedge\bigwedge_{j=1}^{i-1}\neg\mu_j^{\mathbf{A}}$, and then invokes a minimization procedure on $\eta_i^{\mathbf{A}}$ to compute a partial truth assignment $\mu_i^{\mathbf{A}}$ satisfying properties (i), (ii) and (iii). Then, the solver adds the clause $\neg\mu_i^{\mathbf{A}}$ to ensure property (ii) and it continues the search. This process is iterated until ψ_{N+1} is found to be unsatisfiable for some N, and the set $\{\mu_i^{\mathbf{A}}\}_{i=1}^{N}$ is returned.

The minimization procedure consists in iteratively dropping literals one-by-one from $\eta_i^{\mathbf{A}}$, checking if it still satisfies the formula. The outline of this minimization procedure is shown in Algorithm 1. Each minimization step is $O(\#clauses\cdot\#vars)$.

Algorithm 1 MINIMIZE-ASSIGNMENT$(\psi_i, \eta_i, \mathbf{A})$ // $\psi_i \stackrel{\text{def}}{=} \psi \wedge \bigwedge_{j=1}^{i-1} \neg\mu_j^{\mathbf{A}}$, $\eta_i = \eta_i^{\mathbf{A}} \cup \eta_i^{\mathbf{B}}$

```
1: μ_i^A ← η_i^A
2: for ℓ ∈ μ_i^A do
3:     if ψ_i|_[μ_i^A \ {ℓ} ∪ η_i^B] = ⊤ then
4:         μ_i^A ← μ_i^A \ {ℓ}
5: return μ_i^A
```

Notice that, since we are in the context of *projected* AllSAT, the minimization algorithm only minimizes the relevant variables in $\mathbf{A}$, and the truth value of existentially quantified variables in $\mathbf{B}$ is still used to check the satisfiability of the formula by the current partial assignment. Moreover, to enforce the pairwise disjointness between the assignments, ψ_i in Algorithm 1 refers to the original formula conjoined with all current blocking clauses $\bigwedge_{j=1}^{i-1} \neg\mu_j^{\mathbf{A}}$, whereas conflict clauses are excluded by the minimization, being redundant.

We stress the fact that the work described in this paper is agnostic w.r.t. the disjoint AllSAT procedure used, provided its output assignments match conditions (i)-(iii) above.

3 The impact of CNF transformations for AllSAT

In this section we analyze the impact of different CNF-izations on the AllSAT task. In particular, we focus on $\mathsf{CNF_{Ts}}$ [20] and $\mathsf{CNF_{PG}}$ [16]. We point out how CNF-izing AllSAT problems using these transformations can introduce unexpected drawbacks in the enumeration process. In fact, we show that the resulting encodings can prevent the solver from effectively minimizing the models, and thus from enumerating a small set of short partial models.

3.1 The impact of Tseitin CNF transformation

We show that preprocessing the input formula using the $\mathsf{CNF_{Ts}}$ transformation [20] can be problematic for enumeration. We first illustrate this issue with an example.

▶ **Example 3.** Consider the propositional formula

$$\varphi \stackrel{\text{def}}{=} \overbrace{(A_1 \wedge A_2)}^{B_1} \vee (\overbrace{\underbrace{(\overbrace{(A_3 \vee A_4)}^{B_2} \wedge \overbrace{(A_5 \vee A_6)}^{B_3})}_{B_4} \leftrightarrow A_7)}^{B_5} \tag{1}$$

over the set of atoms $\mathbf{A} \stackrel{\text{def}}{=} \{A_1, A_2, A_3, A_4, A_5, A_6, A_7\}$. We first notice that the minimal partial truth assignment:

$$\mu^{\mathbf{A}} \stackrel{\text{def}}{=} \{\neg A_3, \neg A_4, \neg A_7\} \tag{2}$$

suffices to satisfy φ, even though it does not assign a truth value to the sub-formulas $(A_1 \wedge A_2)$ and $(A_5 \vee A_6)$ since the atoms A_1, A_2, A_5, A_6 are not assigned.

Nevertheless, φ is not in CNF, and thus it must be CNF-ized by the solver before starting

the enumeration process. If $\mathsf{CNF_{Ts}}$ is used, then the following CNF formula is obtained:

$$
\begin{aligned}
&\mathsf{CNF_{Ts}}(\varphi) \stackrel{\text{def}}{=} \\
&(\neg B_1 \vee \ \ A_1) \wedge (\neg B_1 \vee \ \ A_2) \wedge (\ \ B_1 \vee \neg A_1 \vee \neg A_2) \wedge \quad //(B_1 \leftrightarrow (A_1 \wedge A_2)) \qquad (3a)\\
&(\ \ B_2 \vee \neg A_3) \wedge (\ \ B_2 \vee \neg A_4) \wedge (\neg B_2 \vee \ \ A_3 \vee \ \ A_4) \wedge \quad //(B_2 \leftrightarrow (A_3 \vee A_4)) \qquad (3b)\\
&(\ \ B_3 \vee \neg A_5) \wedge (\ \ B_3 \vee \neg A_6) \wedge (\neg B_3 \vee \ \ A_5 \vee \ \ A_6) \wedge \quad //(B_3 \leftrightarrow (A_5 \vee A_6)) \qquad (3c)\\
&(\neg B_4 \vee \ \ B_2) \wedge (\neg B_4 \vee \ \ B_3) \wedge (\ \ B_4 \vee \neg B_2 \vee \neg B_3) \wedge \quad //(B_4 \leftrightarrow (B_2 \wedge B_3)) \qquad (3d)\\
&(\neg B_5 \vee \ \ B_4 \vee \neg A_7) \wedge (\neg B_5 \vee \neg B_4 \vee \ \ A_7) \qquad \wedge \quad //(B_5 \leftrightarrow (B_4 \leftrightarrow A_7)) \qquad (3e)\\
&(\ \ B_5 \vee \ \ B_4 \vee \ \ A_7) \wedge (\ \ B_5 \vee \neg B_4 \vee \neg A_7) \qquad \wedge \\
&(\ \ B_1 \vee \ \ B_5) \qquad \wedge \qquad (3f)
\end{aligned}
$$

The fresh atoms $\mathbf{B} \stackrel{\text{def}}{=} \{B_1, B_2, B_3, B_4, B_5\}$ label sub-formulas as in (1). The solver proceeds to compute $\mathcal{TA}(\exists \mathbf{B}.\mathsf{CNF_{Ts}}(\varphi))$ by enumerating the models of $\mathsf{CNF_{Ts}}(\varphi)$ projected over $\mathbf{A}$. Suppose, e.g., that the solver picks non-deterministic choices, deciding the atoms in the order $\{B_1, A_1, A_2, B_2, A_3, A_4, B_3, A_5, A_6, B_4, B_5, A_7\}$ and branching with negative value first. Then, the first (sorted) total truth assignment found is:

$$
\eta \stackrel{\text{def}}{=} \{\underbrace{\neg B_1, \neg B_2, \neg B_3, \neg B_4, B_5}_{\eta^{\mathbf{B}}}, \underbrace{\neg A_1, \neg A_2, \neg A_3, \neg A_4, \neg A_5, \neg A_6, \neg A_7}_{\eta^{\mathbf{A}}}\} \qquad (4)
$$

which contains $\mu^{\mathbf{A}}$ (2). The minimization procedure looks for a *minimal* subset $\mu^{\mathbf{A}\prime}$ of $\eta^{\mathbf{A}}$ s.t. $\mu^{\mathbf{A}\prime} \cup \eta^{\mathbf{B}} \models \mathsf{CNF_{Ts}}(\varphi)$. One possible output of this procedure is the minimal assignment:

$$
\mu^{\mathbf{A}\prime} \stackrel{\text{def}}{=} \{\neg A_1, \neg A_3, \neg A_4, \neg A_5, \neg A_6, \neg A_7\}. \qquad (5)
$$

We notice that the partial truth assignment $\mu^{\mathbf{A}}$ (2) satisfies φ and it is s.t. $\mu^{\mathbf{A}} \subset \mu^{\mathbf{A}\prime}$, but it *does not satisfy* $\exists \mathbf{B}.\mathsf{CNF_{Ts}}(\varphi)$. In fact, three clauses of $\mathsf{CNF_{Ts}}(\varphi)$ in (3a) and (3c) are not satisfied by $\mu^{\mathbf{A}} \cup \eta^{\mathbf{B}}$, since $\mathsf{CNF_{Ts}}(\varphi)|_{\mu^{\mathbf{A}} \cup \eta^{\mathbf{B}}} = (\neg A_1 \vee \neg A_2) \wedge (\neg A_5) \wedge (\neg A_6)$. We remark that this is not a coincidence, since there is no $\eta^{\mathbf{B}\prime}$ such that $\mu^{\mathbf{A}} \cup \eta^{\mathbf{B}\prime} \models \mathsf{CNF_{Ts}}(\varphi)$, because (3a) and (3c) cannot be satisfied without assigning any atom in $\{A_1, A_2\}$ and $\{A_5, A_6\}$ respectively.

Finding $\mu^{\mathbf{A}\prime}$ (5) instead of $\mu^{\mathbf{A}}$ (2) clearly causes an efficiency problem, since finding longer partial models implies that the total number of enumerated models could be up to exponentially larger. For instance, instead of the single partial assignment $\mu^{\mathbf{A}}$ (2), the solver may return the following list of 9 partial assignments satisfying $\exists \mathbf{B}.\mathsf{CNF_{Ts}}(\varphi)$:

$$
\begin{array}{cccccccc}
\overbrace{}^{B_1} & & & & \overbrace{}^{B_3} & & & \\
\{\neg A_1, & & \neg A_3, & \neg A_4, & \neg A_5, & \neg A_6, & \neg A_7 & \} \quad //\{\neg B_1, \neg B_3\} \\
\{\neg A_1, & & \neg A_3, & \neg A_4, & A_5, & & \neg A_7 & \} \quad //\{\neg B_1, \ B_3\} \\
\{\neg A_1, & & \neg A_3, & \neg A_4, & \neg A_5, & A_6, & \neg A_7 & \} \quad //\{\neg B_1, \ B_3\} \\
\{ A_1, & \neg A_2, & \neg A_3, & \neg A_4, & \neg A_5, & \neg A_6, & \neg A_7 & \} \quad //\{\neg B_1, \neg B_3\} \\
\{ A_1, & \neg A_2, & \neg A_3, & \neg A_4, & A_5, & & \neg A_7 & \} \quad //\{\neg B_1, \ B_3\} \\
\{ A_1, & \neg A_2, & \neg A_3, & \neg A_4, & \neg A_5, & A_6, & \neg A_7 & \} \quad //\{\neg B_1, \ B_3\} \\
\{ A_1, & A_2, & \neg A_3, & \neg A_4, & \neg A_5, & \neg A_6, & \neg A_7 & \} \quad //\{\ B_1, \neg B_3\} \\
\{ A_1, & A_2, & \neg A_3, & \neg A_4, & A_5, & & \neg A_7 & \} \quad //\{\ B_1, \ B_3\} \\
\{ A_1, & A_2, & \neg A_3, & \neg A_4, & \neg A_5, & A_6, & \neg A_7 & \} \quad //\{\ B_1, \ B_3\}
\end{array} \qquad (6)
$$

where $\mu^{\mathbf{A}\prime}$ (5) is the first in the list. ◇

The example above shows an intrinsic problem of $\mathsf{CNF_{Ts}}$ when used for enumeration: *if a minimal partial assignment $\mu^{\mathbf{A}}$ suffices to satisfy φ, this does not imply that $\mu^{\mathbf{A}}$ suffices to satisfy $\exists\mathbf{B}.\mathsf{CNF_{Ts}}(\varphi)$,* i.e., that some $\eta^{\mathbf{B}}$ exists such that $\mu^{\mathbf{A}} \cup \eta^{\mathbf{B}}$satisfies $\mathsf{CNF_{Ts}}(\varphi)$ [17].

In fact, consider a generic non-CNF formula $\varphi(\mathbf{A})$ and a minimal partial truth assignment $\mu^{\mathbf{A}}$ that satisfies φ, and let φ_i be some sub-formula of φ which is not assigned a truth value by $\mu^{\mathbf{A}}$ —for instance, because φ_i occurs into some positive subformula $\varphi_i \vee \varphi_j$ and $\mu^{\mathbf{A}}$ satisfies φ_j. (In Example 3, $\mu^{\mathbf{A}} \stackrel{\text{def}}{=} \{\neg A_3, \neg A_4, \neg A_7\}$, $\varphi_i \stackrel{\text{def}}{=} (A_1 \wedge A_2)$ or $\varphi_i \stackrel{\text{def}}{=} (A_5 \vee A_6)$ respectively.) Then $\mathsf{CNF_{Ts}}$ conjoins to the main formula the definition $(B_i \leftrightarrow \varphi_i)$, so that every satisfying partial truth assignment $\mu^{\mathbf{A}\prime}$ is forced to assign a truth value to φ_i and thus to some of its atoms, which may not occur in $\mu^{\mathbf{A}}$, so that $\mu^{\mathbf{A}\prime} \supset \mu^{\mathbf{A}}$. (In the example, the clauses in (3a) and (3c) force $\mu^{\mathbf{A}\prime}$ to assign a truth value also to $(A_1 \wedge A_2)$ and $(A_5 \vee A_6)$ respectively.)

Thus, by using $\mathsf{CNF_{Ts}}$, instead of enumerating one minimal partial model $\mu^{\mathbf{A}}$ for φ, the solver may be forced to enumerate many partial models $\mu^{\mathbf{A}\prime}$ that are minimal for $\exists\mathbf{B}.\mathsf{CNF_{Ts}}(\varphi)$ but not for φ, so that their number can be up to exponentially larger in the number of unassigned atoms in $\mu^{\mathbf{A}}$. In fact, each such model $\mu^{\mathbf{A}\prime}$ conjoins to $\mu^{\mathbf{A}}$ one of the (up to $2^{|\mathbf{A}|-|\mu^{\mathbf{A}}|}$) partial assignments which are needed to evaluate to either $\top$ or $\bot$ all unassigned φ_i's. (E.g., in (6), the solver enumerates nine $\mu^{\mathbf{A}\prime}$s by conjoining $\mu^{\mathbf{A}}$ (2) with an exhaustive enumeration of partial assignments to A_1, A_2, A_5, A_6 that evaluate $(A_1 \wedge A_2)$ and $(A_5 \vee A_6)$ to either $\top$ or $\bot$.) This may drastically affect the effectiveness of the enumeration.

3.2 The impact of Plaisted and Greenbaum CNF transformation

We point out how the $\mathsf{CNF_{PG}}$ [16] can be used to solve these issues, but only in part. We first illustrate it with an example.

▶ **Example 4.** Consider the formula φ (1) and the minimal satisfying assignment $\mu^{\mathbf{A}}$ (2) as in Example 3. Suppose that φ is converted into CNF using $\mathsf{CNF_{PG}}$. Then, the following CNF formula is obtained:

$$
\begin{aligned}
&\mathsf{CNF_{PG}}(\varphi) \stackrel{\text{def}}{=} \\
&(\neg B_1 \vee \ \ A_1) \wedge (\neg B_1 \vee \ \ A_2) && \wedge && //(B_1 \rightarrow (A_1 \wedge A_2)) && \text{(7a)}\\
&(\ \ B_2 \vee \neg A_3) \wedge (\ \ B_2 \vee \neg A_4) \wedge (\neg B_2 \vee \ \ A_3 \vee \ \ A_4) \wedge && //(B_2 \leftrightarrow (A_3 \vee A_4)) && \text{(7b)}\\
&(\ \ B_3 \vee \neg A_5) \wedge (\ \ B_3 \vee \neg A_6) \wedge (\neg B_3 \vee \ \ A_5 \vee \ \ A_6) \wedge && //(B_3 \leftrightarrow (A_5 \vee A_6)) && \text{(7c)}\\
&(\neg B_4 \vee \ \ B_2) \wedge (\neg B_4 \vee \ \ B_3) \wedge (\ \ B_4 \vee \neg B_2 \vee \neg B_3) \wedge && //(B_4 \leftrightarrow (B_2 \wedge B_3)) && \text{(7d)}\\
&(\neg B_5 \vee \ \ B_4 \vee \neg A_7) \wedge (\neg B_5 \vee \neg B_4 \vee \ \ A_7) && \wedge && //(B_5 \rightarrow (B_4 \leftrightarrow A_7)) && \text{(7e)}\\
&(\ \ B_1 \vee \ \ B_5) && \wedge && && \text{(7f)}
\end{aligned}
$$

We highlight that (7a) and (7e) are shorter than (3a) and (3e) respectively, since the corresponding sub-formulas occur only with positive polarity. Suppose, as in Example 3, that the solver finds the total truth assignment $\eta \stackrel{\text{def}}{=} \eta^{\mathbf{B}} \cup \eta^{\mathbf{A}}$ in (4). In this case, one possible output of the minimization procedure is the minimal partial truth assignment:

$$\mu^{\mathbf{A}\prime\prime} \stackrel{\text{def}}{=} \{\neg A_3, \neg A_4, \neg A_5, \neg A_6, \neg A_7\}. \tag{8}$$

This assignment is a strict sub-assignment of $\mu^{\mathbf{A}\prime}$ in (5), since the atom A_1 is not assigned. This is possible because the sub-formula $(A_1 \wedge A_2)$ is labelled by B_1 using a single implication, and the clause representing $(B_1 \rightarrow (A_1 \wedge A_2))$ is satisfied by $\eta^{\mathbf{B}}(B_1) = \bot$ even without assigning A_1 and A_2. Nevertheless, the assignment $\mu^{\mathbf{A}}$ in (2) that satisfies φ *still does not satisfy* $\exists\mathbf{B}.\mathsf{CNF_{PG}}(\varphi)$.

Indeed, sub-formulas occurring with double polarity are labelled using double implications as for $\mathsf{CNF_{Ts}}$, raising the same problems as the latter. For instance, the sub-formula $(A_5 \vee A_6)$ occurs with double polarity, since it is under the scope of an "$\leftrightarrow$". Hence, the clauses in (7c) must be satisfied by assigning a truth value also to A_5 or A_6, and so the partial truth assignment $\mu^{\mathbf{A}}$ in (2) does not suffice to satisfy $\exists\mathbf{B}.\mathsf{CNF_{PG}}(\varphi)$. $\diamond$

The example above shows that $\mathsf{CNF_{PG}}$ has an advantage over $\mathsf{CNF_{Ts}}$ when enumerating partial assignments, but it overcomes its effectiveness issues only in part, *because a minimal assignment* $\mu^{\mathbf{A}}$ *satisfying* φ *may not suffice to satisfy* $\exists\mathbf{B}.\mathsf{CNF_{PG}}(\varphi)$, as with $\mathsf{CNF_{Ts}}$.

Consider, as in §3.1, a generic non-CNF formula $\varphi(\mathbf{A})$ and a partial truth assignment $\mu^{\mathbf{A}}$ that satisfies φ without assigning a truth value to some sub-formula φ_i. Suppose that φ_i occurs only positively in φ —for the negative case the reasoning is dual. (In Example 4, $\mu^{\mathbf{A}} \stackrel{\text{def}}{=} \{\neg A_3, \neg A_4, \neg A_7\}$, $\varphi_i \stackrel{\text{def}}{=} (A_1 \wedge A_2)$.) Since $\mathsf{CNF_{PG}}$ introduces only the clauses representing $(B_i \rightarrow \varphi_i)$ —and not those representing $(B_i \leftarrow \varphi_i)$— the solver is no longer forced to assign a truth value to φ_i, because it suffices to assign $\eta^{\mathbf{B}}(B_i) = \bot$. (In the example, $(A_1 \wedge A_2)$ is labelled with B_1 in (7a).) In this case, φ_i plays the role of a "don't care" term, and this property allows for the enumeration of shorter partial assignments.

Nevertheless, a sub-formula can be "don't care" only if it occurs with single polarity. In fact, if φ_i occurs with double polarity —as it is the case, e.g., of sub-formulas under the scope of an "$\leftrightarrow$"— then φ_i is labelled with a double implication $(B_i \leftrightarrow \varphi_i)$, yielding the same drawbacks as with $\mathsf{CNF_{Ts}}$. (In the example, $(A_5 \vee A_6)$ occurs with double polarity, and $\mu^{\mathbf{A}\prime}$ is forced to assign a truth value also to A_5 or A_6 to satisfy the clauses in (7c)).

Notice that, to maximize the benefits of $\mathsf{CNF_{PG}}$, the sub-formulas that should be treated as "don't care" must have their label assigned to false. In practice, this can be achieved in part by instructing the solver to split on negative values in decision branches[2]. Even though the solver is not guaranteed to always assign to false the labels of "don't care" sub-formulas, we empirically verified that this heuristic provides a good approximation of this behaviour.

4 Enhancing enumeration via NNF preprocessing

In this section, we propose a possible solution to address the shortcomings of $\mathsf{CNF_{Ts}}$ and $\mathsf{CNF_{PG}}$ CNF-izations in model enumeration, described in §3. We show that a simple preprocessing can avoid this situation. We transform first the input formula into an NNF DAG. In fact, NNF guarantees that each sub-formula occurs only positively, as every sub-formula φ_i occurring with double polarity is converted into two syntactically-different sub-formulas $\varphi_i^+ \stackrel{\text{def}}{=} \mathsf{NNF}(\varphi_i)$ and $\varphi_i^- \stackrel{\text{def}}{=} \mathsf{NNF}(\neg\varphi_i)$ —each occurring only positively— which are then labelled —with single implications— with two distinct atoms B_i^+ and B_i^- respectively. To improve the efficiency of the enumeration procedure without affecting its outcome, we also add the clauses $(\neg B_i^+ \vee \neg B_i^-)$ when both B_i^+ and B_i^- are introduced, which prevent the solver from assigning both B_i^+ and B_i^- to true, and thus from exploring inconsistent search branches.

We remark that even with this preprocessing we produce a linear-size CNF encoding, since the $\mathsf{NNF}(\varphi)$ DAG has linear size w.r.t. φ (see §2.1), and $\mathsf{CNF_{PG}}$ introduces one label definition for each DAG node, each consisting of 1 or 2 clauses. We illustrate the benefit of this additional preprocessing with the following example.

[2] To exploit this heuristic also for sub-formulas occurring only negatively, the latter can be labelled with a negative label $\neg B_i$ as $(\neg B_i \leftarrow \varphi_i)$.

▶ **Example 5.** Consider the formula φ of Example 3. By converting it into NNF, we obtain:

$$\varphi' \stackrel{\text{def}}{=} \overbrace{(A_1 \wedge A_2)}^{B_1} \vee (\overbrace{(\overbrace{(\overbrace{(\neg A_3 \wedge \neg A_4)}^{B_2^-} \vee \overbrace{(\neg A_5 \wedge \neg A_6)}^{B_3^-})}^{B_4^-} \vee A_7)}^{B_5} \wedge \overbrace{(\overbrace{(\overbrace{(A_3 \vee A_4)}^{B_2^+} \wedge \overbrace{(A_5 \vee A_6)}^{B_3^+})}^{B_4^+} \vee \neg A_7)}^{B_6}) \overbrace{\qquad\qquad}^{B_7} \tag{9}$$

Suppose, then, that the formula is converted into CNF using $\mathsf{CNF_{PG}}$. Then, the following CNF formula is obtained:

$$\begin{aligned}
\mathsf{CNF_{PG}}(\mathsf{NNF}(\varphi)) \stackrel{\text{def}}{=} & (\neg B_1 \vee \ A_1) \wedge (\neg B_1 \vee \ A_2) \wedge && //(B_1 \rightarrow (\ A_1 \wedge \ A_2)) && \text{(10a)}\\
& (\neg B_2^- \vee \neg A_3) \wedge (\neg B_2^- \vee \neg A_4) \wedge && //(B_2^- \rightarrow (\neg A_3 \wedge \neg A_4)) && \text{(10b)}\\
& (\neg B_3^- \vee \neg A_5) \wedge (\neg B_3^- \vee \neg A_6) \wedge && //(B_3^- \rightarrow (\neg A_5 \wedge \neg A_6)) && \text{(10c)}\\
& (\neg B_4^- \vee \ B_2^- \vee \ B_3^-) && \wedge \quad //(B_4^- \rightarrow (\ B_2^- \vee \ B_3^-)) && \text{(10d)}\\
& (\neg B_5 \vee \ B_4^- \vee \ A_7) && \wedge \quad //(B_5 \rightarrow (\ B_4^- \vee \ A_7)) && \text{(10e)}\\
& (\neg B_2^+ \vee \ A_3 \vee \ A_4) && \wedge \quad //(B_2^+ \rightarrow (\ A_3 \vee \ A_4)) && \text{(10f)}\\
& (\neg B_3^+ \vee \ A_5 \vee \ A_6) && \wedge \quad //(B_3^+ \rightarrow (\ A_5 \vee \ A_6)) && \text{(10g)}\\
& (\neg B_4^+ \vee \ B_2^+) \wedge (\neg B_4^+ \vee \ B_3^+) \wedge && //(B_4^+ \rightarrow (\ B_2^+ \wedge \ B_3^+)) && \text{(10h)}\\
& (\neg B_6 \vee \ B_4^+ \vee \neg A_7) && \wedge \quad //(B_6 \rightarrow (\ B_4^+ \vee \neg A_7)) && \text{(10i)}\\
& (\neg B_7 \vee \ B_5) \wedge (\neg B_7 \vee \ B_6) \wedge && //(B_7 \rightarrow (\ B_5 \wedge \ B_6)) && \text{(10j)}\\
& (\ B_1 \vee \ B_7) && \wedge && \text{(10k)}\\
& (\neg B_2^+ \vee \neg B_2^-) && \wedge && \text{(10l)}\\
& (\neg B_3^+ \vee \neg B_3^-) && \wedge && \text{(10m)}\\
& (\neg B_4^+ \vee \neg B_4^-) && && \text{(10n)}
\end{aligned}$$

Suppose, e.g., that the solver picks non-deterministic choices, deciding the atoms in the order $\{B_1, A_1, A_2, B_3^-, A_5, A_6, B_2^-, A_3, A_4, B_4^-, B_5, A_7, B_2^+, B_3^+, B_4^+, B_6, B_7\}$, branching with a negative value first. Then, the first total truth assignment found is:

$$\eta \stackrel{\text{def}}{=} \{\underbrace{\neg B_1, B_2^-, \neg B_3^-, B_4^-, B_5, \neg B_2^+, \neg B_3^+, \neg B_4^+, B_6, B_7}_{\eta^{\mathbf{B}}}, \underbrace{\neg A_1, \neg A_2, \neg A_3, \neg A_4, \neg A_5, \neg A_6, \neg A_7}_{\eta^{\mathbf{A}}}\}. \tag{11}$$

In this case, the minimization procedure returns $\mu^{\mathbf{A}} \stackrel{\text{def}}{=} \{\neg A_3, \neg A_4, \neg A_7\}$ as in (5), achieving full minimization. With this additional preprocessing, in fact, the solver is no longer forced to assign a truth value to A_5 or A_6. This is possible because, even though $(A_5 \vee A_6)$ occurs with double polarity in φ, in $\mathsf{NNF}(\varphi)$ its positive and negative occurrences are converted into $(A_5 \vee A_6)$ and $(\neg A_5 \wedge \neg A_6)$ respectively. Since they appear as two syntactically-different sub-formulas, $\mathsf{CNF_{PG}}$ labels them —with single implications— using two different atoms B_3^+ and B_3^- respectively. This allows the solver to find a model η that assigns both B_3^- and B_3^+ to false. Hence, the clauses in (10c) and (10g) are satisfied even without assigning A_5 and A_6, and thus these atoms can be dropped by the minimization procedure. ◇

The key idea behind this additional preprocessing is that each sub-formula of $\mathsf{NNF}(\varphi)$ occurs only positively, so that $\mathsf{CNF_{PG}}$ labels them with single implications, and the solver is no longer forced to assign them a truth value. Consider a sub-formula φ_i that occurs with double polarity in φ. In $\mathsf{NNF}(\varphi)$ the two subformulas $\varphi_i^+ \stackrel{\text{def}}{=} \mathsf{NNF}(\varphi_i)$ and $\varphi_i^- \stackrel{\text{def}}{=} \mathsf{NNF}(\neg\varphi_i)$ occur only positively. Then, instead of adding $(B_i \leftrightarrow \varphi_i)$, we add $(B_i^+ \rightarrow \varphi_i^+) \wedge (B_i^- \rightarrow \varphi_i^-)$, and the solver can find a truth assignment η that assigns both B_i^- and B_i^+ to false. (In Example 5, instead of $(B_3 \leftrightarrow (A_5 \vee A_6))$ we add $(B_3^+ \rightarrow (A_5 \vee A_6))$ and $(B_3^- \rightarrow (\neg A_5 \wedge \neg A_6))$.) Thus, the clauses deriving from φ_i can be satisfied even without assigning a truth value to φ_i, whose atoms can be dropped by the minimization procedure —provided that they are not forced to be assigned by some other sub-formula of φ. (In the example, by setting $\eta^{\mathbf{B}}(B_3^+) = \eta^{\mathbf{B}}(B_3^+) = \bot$, the clauses in (10c) and (10g) are satisfied even without assigning A_5 and A_6.)

We have the following general fact: *every partial model $\mu^{\mathbf{A}}$ for φ is also a model for* $\exists\mathbf{B}.\mathsf{CNF_{PG}}(\mathsf{NNF}(\varphi))$, that is, if $\mu^{\mathbf{A}} \models \varphi$, then there exists $\eta^{\mathbf{B}}$ s.t. $\mu^{\mathbf{A}} \cup \eta^{\mathbf{B}} \models \mathsf{CNF_{PG}}(\mathsf{NNF}(\varphi))$. (The vice versa holds trivially.) This is illustrated by the following theorem.

▶ **Theorem 6.** *For every partial truth assignment $\mu^{\mathbf{A}}$ s.t. $\mu^{\mathbf{A}} \models \varphi(\mathbf{A})$, there exists a total truth assignment $\eta^{\mathbf{B}}$ s.t. $\mu^{\mathbf{A}} \cup \eta^{\mathbf{B}} \models \mathsf{CNF_{PG}}(\mathsf{NNF}(\varphi))$.*

The proof of this theorem is shown in Appendix C.

We stress the fact that this does not guarantee that the enumeration procedure always finds this $\eta^{\mathbf{B}}$, but only that such $\eta^{\mathbf{B}}$ exists. E.g., the enumeration procedure of §2.2 finds a total truth assignment $\eta^{\mathbf{A}} \cup \eta^{\mathbf{B}}$ that satisfies the formula, and then finds $\mu^{\mathbf{A}} \subseteq \eta^{\mathbf{A}}$ that is minimal w.r.t. that specific $\eta^{\mathbf{B}}$ so that $\mu^{\mathbf{A}} \cup \eta^{\mathbf{B}} \models \mathsf{CNF_{PG}}(\mathsf{NNF}(\varphi))$, thus the $\eta^{\mathbf{B}}$ found is not guaranteed to be the one that allows for the most effective minimization of $\mu^{\mathbf{A}}$. Ad-hoc enumeration heuristics should be investigated.

Notice that the above fact is agnostic w.r.t. the disjoint-AllSAT algorithm adopted, provided its output assignments match conditions (i)-(iii) in §2.2.

▶ Remark 7. We notice that the pre-conversion into NNF is typically never used in plain SAT *solving*, because it causes the unnecessary duplication of labels B_i^+ and B_i^-, with extra overhead and no benefit for the solver.

5 Experimental evaluation

In this section, we experimentally evaluate the impact of different CNF-izations on the AllSAT task. In order to compare them on a fair ground, we have implemented a base version of each from scratch in PySMT [7], avoiding specific optimizations done by the solvers. We used MathSAT [6] as a SAT enumerator, because it implements the enumeration strategy by [12] described in §2.2. We set the options -dpll.branching_initial_phase=0 to split on the false branch first and -dpll.branching_cache_phase=2 to enable phase caching.

Experiments run on an Intel Xeon Gold 6238R @ 2.20GHz 28 Core machine with 128 GB of RAM and running Ubuntu Linux 20.04. For each instance, we set a timeout of 1200s.

5.1 Datasets description

We consider three sets of benchmarks of non-CNF formulas coming from different sources, both synthetic and real-world. In the first set of benchmarks, we generate random Boolean formulas by nesting Boolean operators up to a fixed depth. The second dataset consists of Boolean formulas encoding properties of ISCAS'85 circuits [4, 8, 19]. As a third set of problems, we consider formulas encoding Booleanized Weighted Model Integration (WMI) problems [13, 14, 18].

The synthetic benchmarks

The synthetic benchmarks are generated by nesting Boolean operators $\wedge, \vee, \leftrightarrow$ until some fixed depth d. Internal and leaf nodes are negated with 50% probability. Operators in internal nodes are chosen randomly, giving less probability to the $\leftrightarrow$ operator. In particular, $\leftrightarrow$ is chosen with a probability of 10%, whereas the other two are chosen with an equal probability of 45%. We generated 100 synthetic instances over a set of 20 Boolean atoms and depth $d = 8$.

The circuits benchmarks

The ISCAS'85 benchmarks are a set of 10 combinatorial circuits used in test generation, timing analysis, and technology mapping [4]. They have well-defined, high-level structures and functions based on multiplexers, ALUs, decoders, and other common building blocks [8]. We generated random instances as described in [19]. In particular, for each circuit, we constrained 60%, 70%, 80%, 90% and 100% of the outputs to either 0 or 1, for a total of 250 instances.

The WMI benchmarks

WMI problems are generated using the procedure described in [18]. Specifically, the paper addresses the problem of enumerating all the different paths of the weight function by encoding it into a skeleton formula. Each instance consists of a skeleton formula of a randomly-generated weight function, where the conditions are only over Boolean atoms. Since the conditions are non-atomic, the resulting formula is not in CNF, and thus we preprocess it with the different CNF-izations before enumerating its models. We generate 10 instances for each depth value 3, 5, 7, 9, each instance involving 10 Boolean atoms and no real variable, for a total of 40 problems.

We remark on two aspects of these benchmarks. First, we have chosen to have Boolean-only weight conditions in order to better analyse the capacity of Boolean reasoning of the solver with the different transformations, without additional factors brought by the SMT component. Nevertheless, we expect to have similar outcomes also for formulas involving both Boolean and SMT($\mathcal{LRA}$) atoms. Notice that these can still be meaningful WMI instances, as the $\mathcal{LRA}$ component may be constrained by the rest of the formula. Second, these formulas contain existentially quantified SMT($\mathcal{EUF}$) atoms, so that we enumerate $\exists \mathbf{y}.\varphi(\mathbf{A}, \mathbf{y})$ by projecting the models of φ over the relevant atoms $\mathbf{A}$ [18].

5.2 Results

Figures 1, 2, and 3 show the results of the experiments on the synthetic, ISCAS'85 and WMI benchmarks, respectively. For each group of benchmarks, we report a set of scatter plots to compare $\mathsf{CNF}_{\mathsf{Ts}}$, $\mathsf{CNF}_{\mathsf{PG}}$ and $\mathsf{NNF} + \mathsf{CNF}_{\mathsf{PG}}$ in terms of number models, in the first row, and execution time, in the second row. Notice the logarithmic scale of the axes!

In §5.3 we also report the CDF of the execution time for plain SAT solving on the same group of benchmarks. We see from the results that, unlike with enumeration, the pre-conversion into NNF has no benefit for plain solving, as we observed in Remark 7.

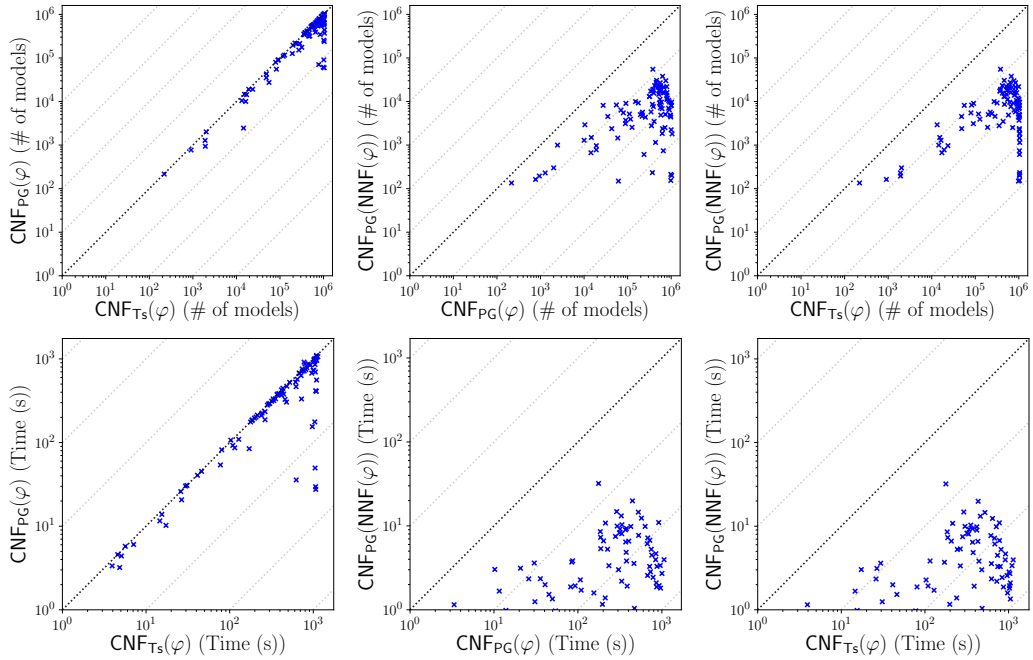


Figure 1 Set of scatter plots comparing the different CNF-izations on the synthetic benchmarks. The first and second rows compare them in terms of number of models and execution time, respectively. All the axes are on a logarithmic scale. In these problems, there were no timeouts.

The synthetic benchmarks

The results on the synthetic benchmarks are shown in Figure 1. All the problems were solved for all the encodings within the timeout. The plots show that $\mathsf{CNF_{PG}}$ performs better than $\mathsf{CNF_{Ts}}$, since it enumerates fewer models (first row) in less time (second row) on every instance. Furthermore, the combination of NNF and $\mathsf{CNF_{PG}}$ yields by far the best results, drastically reducing the number of models and the execution time by orders of magnitude w.r.t. both $\mathsf{CNF_{Ts}}$ and $\mathsf{CNF_{PG}}$.

The circuits benchmarks

Figure 2 shows the performance of the different CNF-izations in the circuits benchmarks. The timeouts are represented by the points on the dashed lines. First, we notice that $\mathsf{CNF_{Ts}}$ and $\mathsf{CNF_{PG}}$ have very similar behaviour, both in terms of execution time and number of models. The reason is that in circuits, it is typical to have a lot of sharing of sub-formulas. Since we constrain the outputs to be 0 or 1 at random [19], most of the sub-formulas occur with double polarity, so that the two encodings are very similar, if not identical. Second, we notice that by converting the formula into NNF before applying $\mathsf{CNF_{PG}}$ the enumeration is much more effective, as a much smaller number of models is enumerated, with only a few outliers. The fact that for some instances $\mathsf{NNF} + \mathsf{CNF_{PG}}$ takes a little more time can be caused by the fact that it can produce a formula that is up to twice as large and contains up to twice as many label atoms as the other two encodings, increasing the time to find the assignments. Notice also that, even enumerating a smaller number of models at a price of a small time-overhead can be beneficial in many applications, for instance in WMI [13, 14, 18].

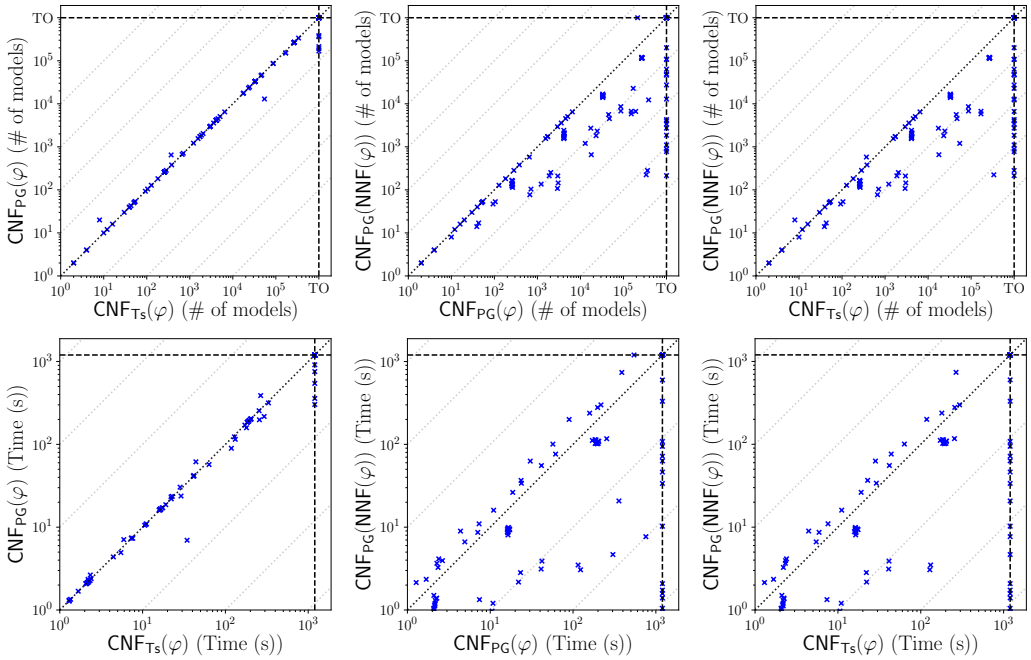


Figure 2 Set of scatter plots comparing the different CNF-izations on the circuits benchmarks. The first and second rows compare them in terms of number of models and execution time, respectively. All the axes are on a logarithmic scale. In these problems, the $\mathsf{CNF_{Ts}}$, $\mathsf{CNF_{PG}}$ and $\mathsf{NNF} + \mathsf{CNF_{PG}}$ reported 49, 44 and 27 timeouts, respectively, represented by the points on the dashed lines.

The WMI benchmarks

The plots in Figure 3 compare the different CNF-izations in the WMI benchmarks in terms of number of models and time. All the problems were solved for all the encodings within the timeout. In these benchmarks, most of the sub-formulas occur with double polarity, so that $\mathsf{CNF_{Ts}}$ and $\mathsf{CNF_{PG}}$ encodings are almost identical, and they obtain very similar results in both metrics. The advantage is significant, instead, if the formula is converted into NNF upfront, since by using $\mathsf{NNF} + \mathsf{CNF_{PG}}$ the solver enumerates a smaller number of models. In this application, it is crucial to enumerate as few models as possible, since for each model an integral must be computed, which is a very expensive operation [13, 14, 18].

5.3 Comparing the CNF encodings for SAT solving

In order to confirm the statement in Remark 7, in the CDFs in Figure 4 we compare the different CNF encodings for *plain SAT solving* on the same benchmarks. Even though these problems are very small for plain solving and SAT solvers deal with them very efficiently, we can see that converting the formula into NNF before applying $\mathsf{CNF_{PG}}$ brings no advantage, and solving is uniformly slower than with $\mathsf{CNF_{PG}}$ or $\mathsf{CNF_{Ts}}$. This shows that our novel technique works specifically for enumeration but not for solving, as expected.

6 Conclusions and future work

We have presented a theoretical and empirical analysis of the impact of different CNF-ization approaches on SAT enumeration. We have shown how the most popular transformations

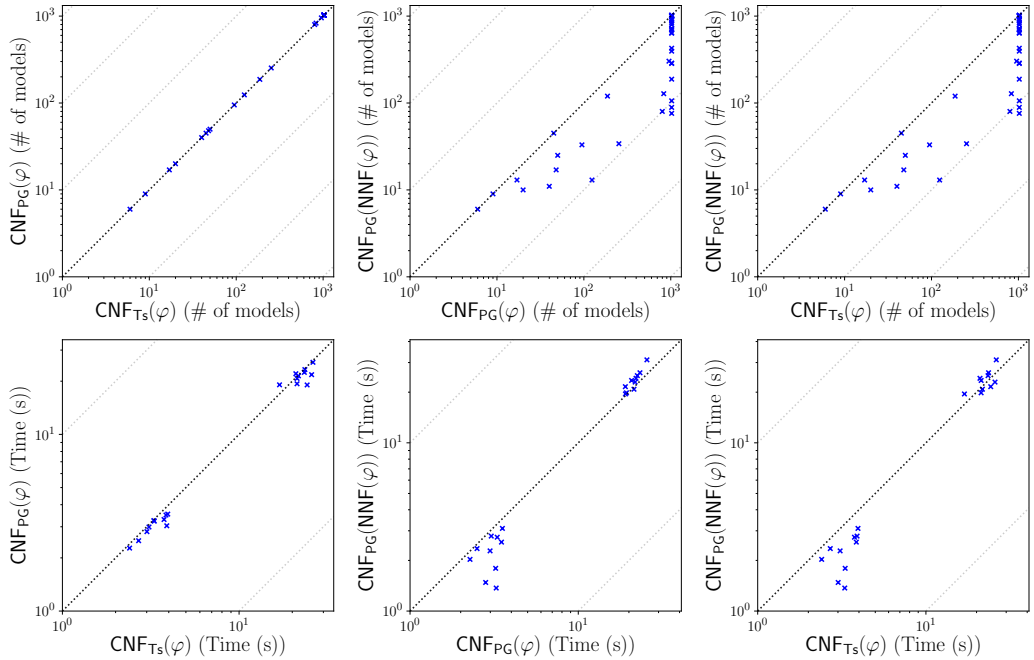
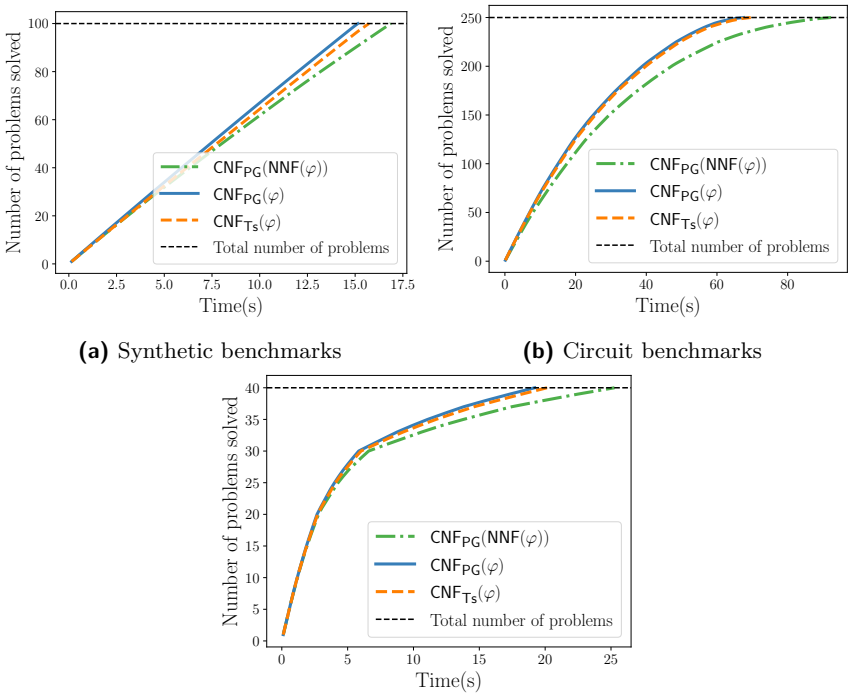


Figure 3 Set of scatter plots comparing the different CNF-izations on the WMI benchmarks. The first and second rows compare them in terms of number of models and execution time, respectively. All the axes are on a logarithmic scale. In these problems, there were no timeouts.

conceived for SAT solving, namely the Tseitin and the Plaisted and Greenbaum CNF-izations, prevent the solver from producing short partial assignments, thus seriously affecting the effectiveness of the enumeration. To overcome this limitation, we have proposed to preprocess the formula by converting it into NNF before applying the Plaisted and Greenbaum transformation. We have shown, both theoretically and empirically, that the latter approach can fully overcome the problem and can drastically reduce both the number of partial assignments and the execution time.

As future research directions, we plan to further investigate the impact of CNF conversion also on disjoint SMT enumeration. We expect that in this domain the impact can be even more relevant, since in SMT multiple instances of the same theory atoms are typically rarer than for atoms in the Boolean case. Also, disjoint SMT enumeration has a fundamental role in Weighted Model Integration [13, 14, 18], an important framework for probabilistic inference in hybrid domains. Hence, we believe that our contribution can have a great impact on this application, where non-CNF formulas occur frequently. Finally, we think that work should be done to understand the impact on enumeration with repetitions, i.e. where models may not be disjoint, for instance in Predicate Abstraction [12].



(a) Synthetic benchmarks

(b) Circuit benchmarks

(c) WMI benchmarks

Figure 4 CDF of the time taken for plain SAT solving using the different CNF transformations. The y-axis reports the instances for which the enumeration finished within the cumulative time on the x-axis.

References

1 A. Biere and H. van Maaren. *Handbook of Satisfiability: Second Edition*. IOS Press, May 2021.

2 Magnus Björk. Successful SAT Encoding Techniques. *Journal on Satisfiability, Boolean Modeling and Computation*, 7(4):189–201, July 2009.

3 Thierry Boy de la Tour. An Optimality Result for Clause Form Translation. *Journal of Symbolic Computation*, 14(4):283–301, October 1992.

4 F. Brglez and H. Fujiwara. A Neutral Netlist of 10 Combinational Benchmark Circuits and a Target Translator in Fortran. In *Proceedings of IEEE International Symposium Circuits and Systems (ISCAS 85)*, pages 677–692. IEEE Press, Piscataway, N.J., 1985.

5 Dmitry Chistikov, Rayna Dimitrova, and Rupak Majumdar. Approximate Counting in SMT and Value Estimation for Probabilistic Programs. In *Proceedings of the 21st International Conference on Tools and Algorithms for the Construction and Analysis of Systems - Volume 9035*, pages 320–334, New York, NY, USA, 2015. Springer-Verlag New York, Inc.

6 Alessandro Cimatti, Alberto Griggio, Bastiaan Joost Schaafsma, and Roberto Sebastiani. The MathSAT5 SMT Solver. In *Tools and Algorithms for the Construction and Analysis of Systems*, Lecture Notes in Computer Science, pages 93–107. Springer, 2013.

7 Marco Gario and Andrea Micheli. PySMT: a solver-agnostic library for fast prototyping of SMT-based algorithms. In *SMT Workshop 2015*, 2015.

8 M.C. Hansen, H. Yalcin, and J.P. Hayes. Unveiling the ISCAS-85 benchmarks: a case study in reverse engineering. *IEEE Design & Test of Computers*, 16(3):72–80, 1999.

9 Markus Iser, Carsten Sinz, and Mana Taghdiri. Minimizing Models for Tseitin-Encoded SAT Instances. In *Theory and Applications of Satisfiability Testing – SAT 2013*, volume 7962, pages 224–232. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013. Series Title: LNCS.

10 Paul Jackson and Daniel Sheridan. The Optimality of a Fast CNF Conversion and its Use with SAT. In *Theory and Applications of Satisfiability Testing – SAT 2004*, 2004.

11 Elias Kuiter, Sebastian Krieter, Chico Sundermann, Thomas Thüm, and Gunter Saake. Tseitin or not Tseitin? The Impact of CNF Transformations on Feature-Model Analyses. In *37th IEEE/ACM Int. Conference on Automated Software Engineering*, pages 1–13. ACM, 2022.

12 Shuvendu K. Lahiri, Robert Nieuwenhuis, and Albert Oliveras. SMT Techniques for Fast Predicate Abstraction. In *Computer Aided Verification*, volume 4144 of *Lecture Notes in Computer Science*, pages 424–437. Springer Berlin Heidelberg, 2006.

13 Paolo Morettin, Andrea Passerini, and Roberto Sebastiani. Efficient Weighted Model Integration via SMT-Based Predicate Abstraction. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 720–728, 2017.

14 Paolo Morettin, Andrea Passerini, and Roberto Sebastiani. Advanced SMT techniques for Weighted Model Integration. *Artificial Intelligence*, 275(C):1–27, 2019.

15 Sibylle Möhle, Roberto Sebastiani, and Armin Biere. Four Flavors of Entailment. In *Theory and Applications of Satisfiability Testing - SAT 2020*, Lecture Notes in Computer Science, pages 62–71. Springer, 2020.

16 David A. Plaisted and Steven Greenbaum. A Structure-preserving Clause Form Translation. *Journal of Symbolic Computation*, 2(3):293–304, 1986.

17 Roberto Sebastiani. Are You Satisfied by This Partial Assignment?, 2020. `arXiv:2003.04225`.

18 Giuseppe Spallitta, Gabriele Masina, Paolo Morettin, Andrea Passerini, and Roberto Sebastiani. SMT-based Weighted Model Integration with Structure Awareness. In *Proceedings of the 38th Conference on Uncertainty in Artificial Intelligence*, volume 180, pages 1876–1885, 2022.

19 Abraham Temesgen Tibebu and Goerschwin Fey. Augmenting All Solution SAT Solving for Circuits with Structural Information. In *IEEE 21st International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS)*, pages 117–122, 2018.

20 G. S. Tseitin. On the complexity of derivation in propositional calculus. In *Automation of Reasoning 2: Classical Papers on Computational Logic 1967-70*, pages 466–483. Springer, 1983.

A Proof for Property 1

We present the proof for Property 1 in Section 2.1.

Proof. The NNF DAG that represents $\mathsf{NNF}(\varphi)$ is a sub-graph of the 2-root DAG for the pair $\langle \mathsf{NNF}(\varphi), \mathsf{NNF}(\neg\varphi)\rangle$. We prove that the latter grows linearly in size w.r.t. φ by reasoning inductively on the structure of φ. (The size of a DAG "..." is denoted with "$|\ldots|$".)
if φ is an atom: then $\mathsf{NNF}(\varphi) = \varphi$ and $\mathsf{NNF}(\neg\varphi) = \neg\varphi$, so that $|\langle \mathsf{NNF}(\varphi), \mathsf{NNF}(\neg\varphi)\rangle| = 2$.
if $\varphi \stackrel{\text{def}}{=} \neg\varphi_1$: we assume by induction that we have computed $\langle \mathsf{NNF}(\varphi_1), \mathsf{NNF}(\neg\varphi_1)\rangle$. Then $\langle \mathsf{NNF}(\varphi), \mathsf{NNF}(\neg\varphi)\rangle = \langle \mathsf{NNF}(\neg\varphi_1), \mathsf{NNF}(\varphi_1)\rangle$ (i.e., we just invert the order of the pair), so that $|\langle \mathsf{NNF}(\varphi), \mathsf{NNF}(\neg\varphi)\rangle| = |\langle \mathsf{NNF}(\neg\varphi_1), \mathsf{NNF}(\varphi_1)\rangle| = |\langle \mathsf{NNF}(\varphi_1), \mathsf{NNF}(\neg\varphi_1)\rangle|$.
if $\varphi \stackrel{\text{def}}{=} (\varphi_1 \bowtie \varphi_2)$ s.t. $\bowtie \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$: we assume by induction we have computed $\langle \mathsf{NNF}(\varphi_1), \mathsf{NNF}(\neg\varphi_1)\rangle$ and $\langle \mathsf{NNF}(\varphi_2), \mathsf{NNF}(\neg\varphi_2)\rangle$. Then we show that

$$|\langle \mathsf{NNF}(\varphi), \mathsf{NNF}(\neg\varphi)\rangle| \leq 18 + |\langle \mathsf{NNF}(\varphi_1), \mathsf{NNF}(\neg\varphi_1)\rangle| + |\langle \mathsf{NNF}(\varphi_2), \mathsf{NNF}(\neg\varphi_2)\rangle|. \quad (12)$$

Specifically:
if $\bowtie$ is $\wedge$: the DAGs for $\mathsf{NNF}(\varphi)$ and $\mathsf{NNF}(\neg\varphi)$ add 2 "$\wedge/\vee$" nodes and $2+2$ arcs:
$\mathsf{NNF}(\quad \varphi_1 \wedge \varphi_2\ ) \stackrel{\text{def}}{=} \mathsf{NNF}(\quad \varphi_1) \wedge \mathsf{NNF}(\quad \varphi_2)$;
$\mathsf{NNF}(\neg(\varphi_1 \wedge \varphi_2)) \stackrel{\text{def}}{=} \mathsf{NNF}(\neg\varphi_1) \vee \mathsf{NNF}(\neg\varphi_2)$.
Thus, $|\langle \mathsf{NNF}(\varphi), \mathsf{NNF}(\neg\varphi)\rangle| = 6 + |\langle \mathsf{NNF}(\varphi_1), \mathsf{NNF}(\neg\varphi_1)\rangle| + |\langle \mathsf{NNF}(\varphi_2), \mathsf{NNF}(\neg\varphi_2)\rangle|$.
if $\bowtie$ is $\vee$: the DAGs for $\mathsf{NNF}(\varphi)$ and $\mathsf{NNF}(\neg\varphi)$ add 2 "$\wedge/\vee$" nodes and $2+2$ arcs:
$\mathsf{NNF}(\quad \varphi_1 \vee \varphi_2\ ) \stackrel{\text{def}}{=} \mathsf{NNF}(\quad \varphi_1) \vee \mathsf{NNF}(\quad \varphi_2)$;
$\mathsf{NNF}(\neg(\varphi_1 \vee \varphi_2)) \stackrel{\text{def}}{=} \mathsf{NNF}(\neg\varphi_1) \wedge \mathsf{NNF}(\neg\varphi_2)$.
Thus, $|\langle \mathsf{NNF}(\varphi), \mathsf{NNF}(\neg\varphi)\rangle| = 6 + |\langle \mathsf{NNF}(\varphi_1), \mathsf{NNF}(\neg\varphi_1)\rangle| + |\langle \mathsf{NNF}(\varphi_2), \mathsf{NNF}(\neg\varphi_2)\rangle|$.
if $\bowtie$ is $\rightarrow$: the DAGs for $\mathsf{NNF}(\varphi)$ and $\mathsf{NNF}(\neg\varphi)$ add 2 "$\wedge/\vee$" nodes and $2+2$ arcs:
$\mathsf{NNF}(\quad \varphi_1 \rightarrow \varphi_2\ ) \stackrel{\text{def}}{=} \mathsf{NNF}(\neg\varphi_1) \vee \mathsf{NNF}(\quad \varphi_2)$;
$\mathsf{NNF}(\neg(\varphi_1 \rightarrow \varphi_2)) \stackrel{\text{def}}{=} \mathsf{NNF}(\quad \varphi_1) \wedge \mathsf{NNF}(\neg\varphi_2)$.
Thus, $|\langle \mathsf{NNF}(\varphi), \mathsf{NNF}(\neg\varphi)\rangle| = 6 + |\langle \mathsf{NNF}(\varphi_1), \mathsf{NNF}(\neg\varphi_1)\rangle| + |\langle \mathsf{NNF}(\varphi_2), \mathsf{NNF}(\neg\varphi_2)\rangle|$.
if $\bowtie$ is $\leftrightarrow$: the DAGs for $\mathsf{NNF}(\varphi)$ and $\mathsf{NNF}(\neg\varphi)$ share the sub-DAGs for $\mathsf{NNF}(\varphi_1)$, $\mathsf{NNF}(\neg\varphi_1)$, $\mathsf{NNF}(\varphi_2)$, $\mathsf{NNF}(\neg\varphi_2)$, adding 3+3 "$\wedge$"/"$\vee$" nodes and $6+6$ arcs:
$\mathsf{NNF}(\quad \varphi_1 \leftrightarrow \varphi_2\ ) \stackrel{\text{def}}{=} (\mathsf{NNF}(\neg\varphi_1) \vee \mathsf{NNF}(\quad \varphi_2)) \wedge (\mathsf{NNF}(\quad \varphi_1) \vee \mathsf{NNF}(\neg\varphi_2))$;
$\mathsf{NNF}(\neg(\varphi_1 \leftrightarrow \varphi_2)) \stackrel{\text{def}}{=} (\mathsf{NNF}(\quad \varphi_1) \vee \mathsf{NNF}(\quad \varphi_2)) \wedge (\mathsf{NNF}(\neg\varphi_1) \vee \mathsf{NNF}(\neg\varphi_2))$.
Thus, $|\langle \mathsf{NNF}(\varphi), \mathsf{NNF}(\neg\varphi)\rangle| = 18 + |\langle \mathsf{NNF}(\varphi_1), \mathsf{NNF}(\neg\varphi_1)\rangle| + |\langle \mathsf{NNF}(\varphi_2), \mathsf{NNF}(\neg\varphi_2)\rangle|$.
Therefore, overall, from (12) we have that $|\langle \mathsf{NNF}(\varphi), \mathsf{NNF}(\neg\varphi)\rangle|$ is $O(|\varphi|)$. ◀

B Proof for Property 2

$\varphi_1\|_\mu$	$\top$	$\top$	$\top$	$*$	$*$	$*$	$\bot$	$\bot$	$\bot$
$\varphi_2\|_\mu$	$\top$	$*$	$\bot$	$\top$	$*$	$\bot$	$\top$	$*$	$\bot$
$\neg(\varphi_1\|_\mu)$	$\bot$	$\bot$	$\bot$	$*$	$*$	$*$	$\top$	$\top$	$\top$
$\varphi_1\|_\mu \wedge \varphi_2\|_\mu$	$\top$	$*$	$\bot$	$*$	$*$	$\bot$	$\bot$	$\bot$	$\bot$
$\varphi_1\|_\mu \vee \varphi_2\|_\mu$	$\top$	$\top$	$\top$	$\top$	$*$	$*$	$\top$	$*$	$\bot$
$\varphi_1\|_\mu \rightarrow \varphi_2\|_\mu$	$\top$	$*$	$\bot$	$\top$	$*$	$*$	$\top$	$\top$	$\top$
$\varphi_1\|_\mu \leftrightarrow \varphi_2\|_\mu$	$\top$	$*$	$\bot$	$*$	$*$	$*$	$\bot$	$*$	$\top$

Figure 5 Three-value-semantics of $\varphi|_\mu$ in terms of $\{\top, \bot, *\}$ ("true", "false", "unknown").

In the following the symbol "$*$" denotes any formula which is not in $\{\top, \bot\}$. Following [17], we adopt a 3-value semantics for residuals $\varphi|_\mu \in \{\top, \bot, *\}$, so that "$\varphi|_\mu = *$" means "$\varphi|_\mu \notin \{\top, \bot\}$" and "$\varphi_1|_\mu = \varphi_2|_\mu$" means that the two residuals $\varphi_1|_\mu$ and $\varphi_2|_\mu$ are either both $\top$, or both $\bot$, or neither is in $\{\top, \bot\}$. (Notice that, in the latter case, $\varphi_1|_\mu = \varphi_2|_\mu$ even when $\varphi_1|_\mu$ and $\varphi_2|_\mu$ are different formulas.) We extend this definition to tuples in an obvious way: $\langle \varphi_1|_\mu, \ldots, \varphi_n|_\mu \rangle = \langle \psi_1|_\mu, \ldots, \psi_n|_\mu \rangle$ iff $\varphi_i|_\mu = \psi_i|_\mu$ for each $i \in [1 \ldots n]$.

The 3-value semantics of the Boolean operators is reported for convenience in Figure 5. As a straightforward consequence of the above semantics, we have that:

$$\begin{aligned} (\neg\varphi)|_\mu &= \neg(\varphi|_\mu) \ \text{ (hereafter simply "}\neg\varphi|_\mu\text{")} \\ (\varphi_1 \bowtie \varphi_2)|_\mu &= \varphi_1|_\mu \bowtie \varphi_2|_\mu \ \text{ for } \bowtie \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}. \end{aligned}$$

Also, the usual transformations apply: $\neg(\varphi_1 \wedge \varphi_2)|_\mu = \neg\varphi_1|_\mu \vee \neg\varphi_2|_\mu$, $\neg(\varphi_1 \vee \varphi_2)|_\mu = \neg\varphi_1|_\mu \wedge \neg\varphi_2|_\mu$, $(\varphi_1 \leftrightarrow \varphi_2)|_\mu = (\neg\varphi_1|_\mu \vee \varphi_2|_\mu) \wedge (\varphi_1|_\mu \vee \neg\varphi_2|_\mu)$ and $\neg(\varphi_1 \leftrightarrow \varphi_2)|_\mu = (\varphi_1|_\mu \vee \varphi_2|_\mu) \wedge (\neg\varphi_1|_\mu \vee \neg\varphi_2|_\mu)$. For convenience, sometimes we denote as $\bar{v}$ the complement of $v \in \{\top, \bot, *\}$, i.e. $\bar{v} \stackrel{\text{def}}{=} \neg v$, so that $\bar{\top} = \bot, \bar{\bot} = \top, \bar{*} = *$.

We prove the following property, from which Property 2 in Section 2.1 follows directly.

▶ **Property 8.** *Consider a Boolean formula $\varphi(\mathbf{A})$, and let $\mathsf{NNF}(\varphi)$ be its NNF DAG. Consider a partial assignment $\mu^{\mathbf{A}}$ on $\mathbf{A}$. Then:*

$$\langle \varphi|_{\mu^{\mathbf{A}}}, \neg\varphi|_{\mu^{\mathbf{A}}} \rangle = \langle \mathsf{NNF}(\varphi)|_{\mu^{\mathbf{A}}}, \mathsf{NNF}(\neg\varphi)|_{\mu^{\mathbf{A}}} \rangle. \tag{13}$$

Proof. As in Appendix A, we prove this fact by reasoning on the 2-root DAG for the pair $\langle \mathsf{NNF}(\varphi), \mathsf{NNF}(\neg\varphi) \rangle$. Specifically, we prove (13) by induction on the structure of φ.

if φ is an atom: then $\mathsf{NNF}(\varphi) = \varphi$ and $\mathsf{NNF}(\neg\varphi) = \neg\varphi$, so that $\varphi|_{\mu^{\mathbf{A}}} = \mathsf{NNF}(\varphi)|_{\mu^{\mathbf{A}}}$ and $\neg\varphi|_{\mu^{\mathbf{A}}} = \mathsf{NNF}(\neg\varphi)|_{\mu^{\mathbf{A}}}$.

if $\varphi \stackrel{\text{def}}{=} \neg\varphi_1$: we assume by induction that $\langle \varphi_1|_{\mu^{\mathbf{A}}}, \neg\varphi_1|_{\mu^{\mathbf{A}}} \rangle = \langle \mathsf{NNF}(\varphi_1)|_{\mu^{\mathbf{A}}}, \mathsf{NNF}(\neg\varphi_1)|_{\mu^{\mathbf{A}}} \rangle$. Let v be s.t. $\langle \varphi|_{\mu^{\mathbf{A}}}, \neg\varphi|_{\mu^{\mathbf{A}}} \rangle = \langle v, \bar{v} \rangle$. Then:

$$\begin{aligned} \langle \neg\varphi_1|_{\mu^{\mathbf{A}}}, \varphi_1|_{\mu^{\mathbf{A}}} \rangle = \langle v, \bar{v} \rangle \Leftrightarrow \\ \langle \varphi_1|_{\mu^{\mathbf{A}}}, \neg\varphi_1|_{\mu^{\mathbf{A}}} \rangle = \langle \bar{v}, v \rangle \stackrel{\text{ind}}{\Leftrightarrow} \\ \langle \mathsf{NNF}(\varphi_1)|_{\mu^{\mathbf{A}}}, \mathsf{NNF}(\neg\varphi_1)|_{\mu^{\mathbf{A}}} \rangle = \langle \bar{v}, v \rangle \Leftrightarrow \\ \langle \mathsf{NNF}(\neg\varphi_1)|_{\mu^{\mathbf{A}}}, \mathsf{NNF}(\varphi_1)|_{\mu^{\mathbf{A}}} \rangle = \langle v, \bar{v} \rangle \end{aligned}$$

if $\varphi \stackrel{\text{def}}{=} (\varphi_1 \bowtie \varphi_2)$**:** s.t. $\bowtie \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$. We assume by induction that

$$\langle \varphi_1|_{\mu^{\mathbf{A}}}, \neg\varphi_1|_{\mu^{\mathbf{A}}} \rangle = \langle \mathsf{NNF}(\varphi_1)|_{\mu^{\mathbf{A}}}, \mathsf{NNF}(\neg\varphi_1)|_{\mu^{\mathbf{A}}} \rangle,$$
$$\langle \varphi_2|_{\mu^{\mathbf{A}}}, \neg\varphi_2|_{\mu^{\mathbf{A}}} \rangle = \langle \mathsf{NNF}(\varphi_2)|_{\mu^{\mathbf{A}}}, \mathsf{NNF}(\neg\varphi_2)|_{\mu^{\mathbf{A}}} \rangle.$$

Then,
if $\bowtie$ **is** $\wedge$**:**

$$\begin{aligned}\langle (\varphi_1 \wedge \varphi_2)|_{\mu^{\mathbf{A}}}, \neg(\varphi_1 \wedge \varphi_2)|_{\mu^{\mathbf{A}}} \rangle &= \langle \varphi_1|_{\mu^{\mathbf{A}}} \wedge \varphi_2|_{\mu^{\mathbf{A}}}, \neg\varphi_1|_{\mu^{\mathbf{A}}} \vee \neg\varphi_2|_{\mu^{\mathbf{A}}} \rangle \stackrel{\text{ind}}{=} \\ \langle \mathsf{NNF}(\varphi_1)|_{\mu^{\mathbf{A}}} \wedge \mathsf{NNF}(\varphi_2)|_{\mu^{\mathbf{A}}}, \mathsf{NNF}(\neg\varphi_1)|_{\mu^{\mathbf{A}}} \vee \mathsf{NNF}(\neg\varphi_2)|_{\mu^{\mathbf{A}}} \rangle &= \\ \langle (\mathsf{NNF}(\varphi_1) \wedge \mathsf{NNF}(\varphi_2))|_{\mu^{\mathbf{A}}}, (\mathsf{NNF}(\neg\varphi_1) \vee \mathsf{NNF}(\neg\varphi_2))|_{\mu^{\mathbf{A}}} \rangle &= \\ \langle \mathsf{NNF}(\varphi_1 \wedge \varphi_2)|_{\mu^{\mathbf{A}}}, \mathsf{NNF}(\neg(\varphi_1 \wedge \varphi_2))|_{\mu^{\mathbf{A}}} \rangle\end{aligned}$$

if $\bowtie$ **is** $\vee$**:**

$$\begin{aligned}\langle (\varphi_1 \vee \varphi_2)|_{\mu^{\mathbf{A}}}, \neg(\varphi_1 \vee \varphi_2)|_{\mu^{\mathbf{A}}} \rangle &= \langle \varphi_1|_{\mu^{\mathbf{A}}} \vee \varphi_2|_{\mu^{\mathbf{A}}}, \neg\varphi_1|_{\mu^{\mathbf{A}}} \wedge \neg\varphi_2|_{\mu^{\mathbf{A}}} \rangle \stackrel{\text{ind}}{=} \\ \langle \mathsf{NNF}(\varphi_1)|_{\mu^{\mathbf{A}}} \vee \mathsf{NNF}(\varphi_2)|_{\mu^{\mathbf{A}}}, \mathsf{NNF}(\neg\varphi_1)|_{\mu^{\mathbf{A}}} \wedge \mathsf{NNF}(\neg\varphi_2)|_{\mu^{\mathbf{A}}} \rangle &= \\ \langle (\mathsf{NNF}(\varphi_1) \vee \mathsf{NNF}(\varphi_2))|_{\mu^{\mathbf{A}}}, (\mathsf{NNF}(\neg\varphi_1) \wedge \mathsf{NNF}(\neg\varphi_2))|_{\mu^{\mathbf{A}}} \rangle &= \\ \langle \mathsf{NNF}(\varphi_1 \vee \varphi_2)|_{\mu^{\mathbf{A}}}, \mathsf{NNF}(\neg(\varphi_1 \vee \varphi_2))|_{\mu^{\mathbf{A}}} \rangle\end{aligned}$$

if $\bowtie$ **is** $\rightarrow$**:**

$$\begin{aligned}\langle (\varphi_1 \rightarrow \varphi_2)|_{\mu^{\mathbf{A}}}, \neg(\varphi_1 \rightarrow \varphi_2)|_{\mu^{\mathbf{A}}} \rangle &= \langle \neg\varphi_1|_{\mu^{\mathbf{A}}} \vee \varphi_2|_{\mu^{\mathbf{A}}}, \varphi_1|_{\mu^{\mathbf{A}}} \wedge \neg\varphi_2|_{\mu^{\mathbf{A}}} \rangle \stackrel{\text{ind}}{=} \\ \langle \mathsf{NNF}(\neg\varphi_1)|_{\mu^{\mathbf{A}}} \vee \mathsf{NNF}(\varphi_2)|_{\mu^{\mathbf{A}}}, \mathsf{NNF}(\varphi_1)|_{\mu^{\mathbf{A}}} \wedge \mathsf{NNF}(\neg\varphi_2)|_{\mu^{\mathbf{A}}} \rangle &= \\ \langle (\mathsf{NNF}(\neg\varphi_1) \vee \mathsf{NNF}(\varphi_2))|_{\mu^{\mathbf{A}}}, (\mathsf{NNF}(\varphi_1) \wedge \mathsf{NNF}(\neg\varphi_2))|_{\mu^{\mathbf{A}}} \rangle &= \\ \langle \mathsf{NNF}(\varphi_1 \rightarrow \varphi_2)|_{\mu^{\mathbf{A}}}, \mathsf{NNF}(\neg(\varphi_1 \rightarrow \varphi_2))|_{\mu^{\mathbf{A}}} \rangle\end{aligned}$$

if $\bowtie$ **is** $\leftrightarrow$**:**

$$\begin{aligned}&\langle (\varphi_1 \leftrightarrow \varphi_2)|_{\mu^{\mathbf{A}}}, \neg(\varphi_1 \leftrightarrow \varphi_2)|_{\mu^{\mathbf{A}}} \rangle = \\ &\langle ((\neg\varphi_1 \vee \varphi_2) \wedge (\varphi_1 \vee \neg\varphi_2))|_{\mu^{\mathbf{A}}}, ((\varphi_1 \vee \varphi_2) \wedge (\varphi_1 \vee \varphi_2))|_{\mu^{\mathbf{A}}} \rangle = \\ &\langle (\neg\varphi_1|_{\mu^{\mathbf{A}}} \vee \varphi_2|_{\mu^{\mathbf{A}}}) \wedge (\varphi_1|_{\mu^{\mathbf{A}}} \vee \neg\varphi_2|_{\mu^{\mathbf{A}}}), (\varphi_1|_{\mu^{\mathbf{A}}} \vee \varphi_2|_{\mu^{\mathbf{A}}}) \wedge (\varphi_1|_{\mu^{\mathbf{A}}} \vee \varphi_2|_{\mu^{\mathbf{A}}}) \rangle \stackrel{\text{ind}}{=} \\ &\langle (\mathsf{NNF}(\neg\varphi_1)|_{\mu^{\mathbf{A}}} \vee \mathsf{NNF}(\varphi_2)|_{\mu^{\mathbf{A}}}) \wedge (\mathsf{NNF}(\varphi_1)|_{\mu^{\mathbf{A}}} \vee \mathsf{NNF}(\neg\varphi_2)|_{\mu^{\mathbf{A}}}), \\ &\quad (\mathsf{NNF}(\varphi_1)|_{\mu^{\mathbf{A}}} \vee \mathsf{NNF}(\varphi_2)|_{\mu^{\mathbf{A}}}) \wedge (\mathsf{NNF}(\varphi_1)|_{\mu^{\mathbf{A}}} \vee \mathsf{NNF}(\varphi_2)|_{\mu^{\mathbf{A}}}) \rangle = \\ &\langle ((\mathsf{NNF}(\neg\varphi_1) \vee \mathsf{NNF}(\varphi_2)) \wedge (\mathsf{NNF}(\varphi_1) \vee \mathsf{NNF}(\neg\varphi_2)))|_{\mu^{\mathbf{A}}}, \\ &\quad ((\mathsf{NNF}(\varphi_1) \vee \mathsf{NNF}(\varphi_2)) \wedge (\mathsf{NNF}(\varphi_1) \vee \mathsf{NNF}(\varphi_2)))|_{\mu^{\mathbf{A}}} \rangle = \\ &\langle \mathsf{NNF}(\varphi_1 \leftrightarrow \varphi_2)|_{\mu^{\mathbf{A}}}, \mathsf{NNF}(\neg(\varphi_1 \leftrightarrow \varphi_2))|_{\mu^{\mathbf{A}}} \rangle\end{aligned}$$

. ◀

Thus, $\langle \varphi|_{\mu^{\mathbf{A}}}, \neg\varphi|_{\mu^{\mathbf{A}}} \rangle = \langle \mathsf{NNF}(\varphi)|_{\mu^{\mathbf{A}}}, \mathsf{NNF}(\neg\varphi)|_{\mu^{\mathbf{A}}} \rangle$ so that $\varphi|_{\mu^{\mathbf{A}}} = v$ iff $\mathsf{NNF}(\varphi)|_{\mu^{\mathbf{A}}} = v$ for every $v \in \{\top, \bot\}$, so that Property 2 in Section 2.1 holds.

C Proof for Theorem 6

We present the proof for Theorem 6 in Section 4.

Proof. We first show how such a $\eta^{\mathbf{B}}$ can be built, then we prove that it satisfies $\mathsf{CNF_{PG}}(\mathsf{NNF}(\varphi))$.
For every sub-formula φ_i of φ, whose positive and negative occurrences in $\mathsf{CNF_{PG}}(\mathsf{NNF}(\varphi))$ are associated with the variables B_i^+ and B_i^- respectively, do:

(a) if $\varphi_i|_{\mu^{\mathbf{A}}} = \top$, and hence $\mathsf{NNF}(\varphi_i)|_{\mu^{\mathbf{A}}} = \top$ and $\mathsf{NNF}(\neg\varphi_i)|_{\mu^{\mathbf{A}}} = \bot$ by Property 2, then set $\eta^{\mathbf{B}}(B_i^+) = \top$ and $\eta^{\mathbf{B}}(B_i^-) = \bot$;

(b) if $\varphi_i|_{\mu^{\mathbf{A}}} = \bot$, and hence $\mathsf{NNF}(\varphi_i)|_{\mu^{\mathbf{A}}} = \bot$ and $\mathsf{NNF}(\neg\varphi_i)|_{\mu^{\mathbf{A}}} = \top$ by Property 2, then set $\eta^{\mathbf{B}}(B_i^+) = \bot$ and $\eta^{\mathbf{B}}(B_i^-) = \top$;

(c) otherwise if $\varphi_i|_{\mu^{\mathbf{A}}} \notin \{\top, \bot\}$, then let $\eta^{\mathbf{B}}(B_i^+) = \eta^{\mathbf{B}}(B_i^-) = \bot$;

(If φ_i occurs only positively or negatively, then we only assign B_i^+ or B_i^- respectively.)
We prove that $\mu^{\mathbf{A}} \cup \eta^{\mathbf{B}} \models \mathsf{CNF_{PG}}(\mathsf{NNF}(\varphi))$ by induction on the structure of $\mathsf{CNF_{PG}}(\mathsf{NNF}(\varphi))$.
Consider a sub-formula φ_i of φ. Then

(i) $\mathsf{CNF_{PG}}(\mathsf{NNF}(\varphi)) \stackrel{\text{def}}{=} \mathsf{CNF_{PG}}(\overbrace{\mathsf{NNF}(\varphi)[\mathsf{NNF}(\varphi_i)|B_i^+]}^{(1)} \wedge \overbrace{(\neg B_i^+ \vee \mathsf{NNF}(\varphi_i))}^{(2)})$ if φ_i occurs positively in φ;

(ii) $\mathsf{CNF_{PG}}(\mathsf{NNF}(\varphi)) \stackrel{\text{def}}{=} \mathsf{CNF_{PG}}(\overbrace{\mathsf{NNF}(\varphi)[\mathsf{NNF}(\neg\varphi_i)|B_i^-]}^{(3)} \wedge \overbrace{(\neg B_i^- \vee \mathsf{NNF}(\neg\varphi_i))}^{(4)})$ if φ_i occurs negatively in φ;

For each pair of cases we have:

(a) **(i)** $\mu^{\mathbf{A}} \cup \eta^{\mathbf{B}} \models (1)$ since we substitute $\top$ for $\top$; $\mu^{\mathbf{A}} \cup \eta^{\mathbf{B}} \models (2)$ since $\mu^{\mathbf{A}} \models \mathsf{NNF}(\varphi_i)$;
(ii) $\mu^{\mathbf{A}} \cup \eta^{\mathbf{B}} \models (3)$ since we substitute $\bot$ for $\bot$; $\mu^{\mathbf{A}} \cup \eta^{\mathbf{B}} \models (4)$ since $\mu^{\mathbf{A}} \models \neg B_i^-$;

(b) **(i)** $\mu^{\mathbf{A}} \cup \eta^{\mathbf{B}} \models (1)$ since we substitute $\bot$ for $\bot$; $\mu^{\mathbf{A}} \cup \eta^{\mathbf{B}} \models (2)$ since $\mu^{\mathbf{A}} \models \neg B_i^+$;
(ii) $\mu^{\mathbf{A}} \cup \eta^{\mathbf{B}} \models (3)$ since we substitute $\top$ for $\top$; $\mu^{\mathbf{A}} \cup \eta^{\mathbf{B}} \models (4)$ since $\mu^{\mathbf{A}} \models \mathsf{NNF}(\neg\varphi_i)$;

(c) **(i)** $\mu^{\mathbf{A}} \cup \eta^{\mathbf{B}} \models (1)$ since we substitute $\bot$ for $*$; $\mu^{\mathbf{A}} \cup \eta^{\mathbf{B}} \models (2)$ since $\mu^{\mathbf{A}} \models \neg B_i^+$;
(ii) $\mu^{\mathbf{A}} \cup \eta^{\mathbf{B}} \models (3)$ since we substitute $\bot$ for $*$; $\mu^{\mathbf{A}} \cup \eta^{\mathbf{B}} \models (4)$ since $\mu^{\mathbf{A}} \models \neg B_i^-$;

Therefore, $\mu^{\mathbf{A}} \cup \eta^{\mathbf{B}} \models \mathsf{CNF_{PG}}(\mathsf{NNF}(\varphi))$. ◀