# Entailment vs. Verification for Partial-assignment Satisfiability and Enumeration

Roberto Sebastiani[0000−0002−0989−6101]

DISI, University of Trento, Italy.
roberto.sebastiani@unitn.it

**Abstract.** Many procedures for SAT-related problems, in particular for those requiring the complete enumeration of satisfying truth assignments, rely their efficiency and effectiveness on the detection of (possibly small) *partial* assignments satisfying an input formula. Surprisingly, there seems to be no unique universally-agreed definition of formula satisfaction by a partial assignment in the literature. In this paper, we analyze in depth the issue of satisfaction by partial assignments, raising a flag about some ambiguities and subtleties of this concept, and investigating their practical consequences. We identify two alternative notions that are implicitly used in the literature, namely *verification* and *entailment*, which coincide if applied to (tautology-free) CNF formulas, but differ and present complementary properties if applied to non-CNF or to existentially-quantified formulas. We show that, although the former is easier to check and as such is implicitly used by most current search procedures, the latter has better theoretical properties, and can improve the efficiency and effectiveness of enumeration procedures.

## 1 Introduction

*Motivations.* Many search procedures for SAT-related problems (e.g., Analytic Tableaux [38], DPLL [14], circuit AllSAT [17]) and many formula compilers (e.g., d-DNNFs [11,13], OBDDs [4] and SDDs [12]) rely their efficiency and effectiveness on the detection of *partial* truth assignments $\mu$ satisfying an input propositional formula $\varphi$, which allows to state that not only $\varphi$ is satisfiable, but also all total assignments extending $\mu$ satisfy $\varphi$. In particular, when it comes to SAT-based problems requiring the *complete enumeration* of satisfying assignments (e.g. #SAT [20], Lazy SMT [34,3], OMT [37], AllSAT and AllSMT [24,6,17], #SMT [31,7], Projected #SAT [2], Projected AllSAT/AllSMT [24,18,40], knowledge compilation into d-DNNF [11,23], satisfiability in modal and description logics [36], Weighted Model Integration [29,39]), the ability of enumerating satisfying partial assignments which are as small as possible is essential, because each of them avoids the enumeration of the whole subtree of total assignments extending it, whose size is exponential in the number of unassigned atoms.

We start by raising a very-basic question: *What should we mean by "a partial assignment $\mu$ satisfies a formula $\varphi$"?* We notice that, quite surprisingly and despite its widespread implicit usage in algorithms, there is no unique and universally-agreed definition for formula satisfaction by partial assignments in the literature: most authors do not define it explicitly; a few others define it only when dealing with CNF formulas (e.g. [22]); the very few who define it, adopt one of the following distinct definitions:

"*applying $\mu$ to $\varphi$ makes $\varphi$ true*" (e.g., [10,17]). We say that $\mu$ *verifies* $\varphi$; [1]
"*all total assignments extending $\mu$ satisfy $\varphi$*" (e.g., [19,34]). We say that $\mu$ *entails* $\varphi$.

Notice that this is not simply an issue of the meaning of the word "satisfy": regardless which verb we may use for it (e.g. "satisfy", "entail", "verify", "imply",...), it should be desirable to have a unique and universally-agreed criterion to establish it.

*Contributions.* In this paper, we analyze in depth the notion of *partial-assignment satisfiability*, in particular when dealing with non-CNF and existentially-quantified formulas, raising a flag about the ambiguities and subtleties of this concept, and investigating their consequences. Our contributions are both theoretical and practical.

*Theoretical analysis.* We show, analyze and discuss the following theoretical facts.

($a$) Whereas for (tautology-free[2]) CNF formulas partial-assignment satisfiability can be indifferently be interpreted either as verification or as entailment because in this case the two concepts coincide, *for non-CNF formulas verification is strictly stronger than entailment*, and they have complementary properties.
($b$) Whereas two equivalent formulas are always entailed by the same partial assignments, *they are not always verified by the same partial assignments*.
($c$) Verification checks that the residual $\varphi|_\mu$ of a formula $\varphi$ under a partial assignment $\mu$ —i.e., the formula obtained by applying the assigned truth values from $\mu$ to the atoms in $\varphi$— is *the true formula*, whereas entailment checks it is *a valid formula*. Thus verification can be computationally much cheaper to check than entailment.
($d$) $(a)$-$(c)$ apply also to existentially-quantified formulas, *even those in CNF*.
($e$) $(a)$-$(c)$ apply also to existentially-quantified formulas resulting from CNF-ization.

We stress the fact that $(b)$ (and $(d)$ and $(e)$ consequently) would be an embarrassing fact if we adopted verification as the definition of partial-assignment satisfiability. As such, we champion the idea that the latter should be defined as entailment [19,34,36], and that verification should be considered an easy-to-check sufficient condition for it.

*Practical consequences.* The above facts have the following practical consequences.

1. In satisfiability, we need to find only one total assignment $\eta$ extending $\mu$, so that entailment produces no benefits wrt. verification and is more expensive $(c)$. Also, due to $(a)$, when the input problem is natively in CNF, the distinction between verification and entailment is not relevant, because these two concepts coincide.

Consequently, both the algorithms for satisfiability and those for enumeration with CNF formulas typically use verification as satisfiability criterion for the current partial assignments, because it is much cheaper and easier to implement than entailment. This is the case, e.g., of classical procedures like Analytic Tableaux [38] and DPLL [14,1], or enumeration, counting, or knowledge compilation procedures, e.g. [11,24,18,2,17,23].

[1] The etymology of "to verify" derives from Latin "*verum facere*", meaning "to make true".
[2] A CNF formula is tautology-free if it contains no tautological clause in the form $(l \vee \neg l \vee ...)$. Since tautological clauses can be easily removed upfront by simple preprocessing, hereafter we often assume w.l.o.g. that CNF formulas are tautology-free.

A few notable exceptions are the Dualiza procedure [27] and the procedures we described in [28,30,16]; also OBDDs [4] and SDDs [12] formula compilers implicitly use entailment to prune branches so that to guarantee canonicity (see below and §4).

The scenario changes when we deal with enumeration-based algorithms applied to non-CNF formulas, or to existentially-quantified formulas, or to CNF-ized formulas.

2. Due to $(a)$, a partial assignment may entail a formula without verifying it. Thus, adopting entailment as partial-assignment satisfiability criterion during the search allows for detecting satisfiability earlier than with verification, and thus for producing smaller partial truth assignments.

Although this fact is not very interesting for satisfiability, it may become fundamental for enumeration, because the detection of a satisfying partial assignment avoids the enumeration of the whole subtree of total assignments extending it, whose size is exponential in the number of unassigned atoms: *the earlier satisfiability is detected, the (up to exponentially) fewer assignments are enumerated*.

3. Due to $(d)$, a partial assignment may entail an existentially-quantified formula, even a CNF one, without verifying it. Thus, as with non-CNF formulas $(b)$, adopting entailment as partial-assignment satisfiability criterion during the search allows for detecting satisfiability earlier than with verification.

This is important because in many application domains fundamental operations —like *pre-image computation* in symbolic model checking (see e.g. [5]) or *predicate abstraction* in SW verification (see e.g. [21,24]) or *projected enumeration* (see e.g. [24,18,40]) or *projected model counting* (see e.g. [2]) — require dealing with existentially-quantified formulas and with the enumeration of partial assignments satisfying them.

4. Due to $(e)$, CNF-izing upfront the non-CNF formula with the standard techniques [42,32] does not fix issues $(a)$-$(c)$, since fresh atoms are introduced, and *a partial assignment over the original atoms may entail the existentially-quantified CNF-ized formula without verifying it.*

This is important because it shows that, although verification and entailment coincide for CNF formulas, Fact 2. above cannot be fixed by simply CNF-izing a formula upfront and running a CNF enumeration procedure based on partial-assignment verification, projecting out the fresh variables introduced by the CNF-ization.

5. Due to $(c)$, checking verification is polynomial, whereas checking entailment is co-NP-complete, since it consists of checking the validity of the residual formula.

This is the main argument in favour of verification vs. entailment. We notice, however, that if $\mu$ entails $\varphi$, then the residual of $\varphi$ under $\mu$ is typically much smaller than $\varphi$ with a much smaller search space, since its atoms are only (a subset of) those that are not assigned by $\mu$. Notice also that, when this happens, entailment prevents from enumerating a number of assignments which is exponential in the number of the atoms in the residual. Therefore, unlike with satisfiability, for enumeration the tradeoff may be favourable to entailment checks.

Based on the considerations above, not only we suggest to adopt entailment rather than verification as unique definition of partial-assignment satisfiability, but also we champion its usage inside enumeration-based search procedures.

*Related Work.* This paper is a revised version of a 2020 unpublished paper [35], which at that time was not published due to lack of algorithmic support and empirical evidence. These came afterwards. Based on the theoretical analysis in [35] in combination with the idea of dualized search from [15,27], in [28,30] we proposed novel enumeration and counting procedures based on entailment. Unfortunately we were not able to produce any implementation efficient enough to compete with current enumeration procedures based on verification. Only very recently, in [16] we have enhanced the HALL enumeration tool for circuits from [17] by substituting verification checks with entailment checks for partial-assignment reductions ("generalizations" in [16]), boosting its performance in terms of both time efficiency and size of partial-assignment sets.

In [25] we addressed a problem which is different although related to $(e)$, showing that adopting a form of CNF-ization different from [42,32] improves the efficiency and effectiveness of enumeration. Such encoding, however, does not fix fact $(e)$ (see §3.3).

*Content.* The rest of the paper is organized as follows. §2 provides the necessary notation, terminology, and concepts used in the paper. §3 presents our theoretical analysis: §3.1 introduces verification and entailment for generic propositional formulas and discusses their relative properties; §3.2 lifts the discussion to existentially-quantified formulas; §3.3 discusses the role of CNF-ization in this context; §3.4 analyzes other candidate forms of partial-assignment satisfaction. §4 discusses the practical consequences of entailment vs. verification. §5 provides conclusions and suggestions for future research.

## 2 Background

In this section we introduce the notation and terminology adopted in this paper, and we recall the standard syntax, semantics and basic facts of propositional logics.

*Notation.* In what follows T, F, ? denote the truth values "true", "false" and "unknown" respectively; $\top$, $\bot$ denote the logic constants "true" and "false" respectively; $A$, $B$ denote propositional atoms; $\varphi$, $\psi$ denote propositional formulas; $\mu, \eta, \gamma$ denote truth value assignments. The symbols $\mathbf{A} \stackrel{\text{def}}{=} \{A_1, ..., A_N\}$ and $\mathbf{B} \stackrel{\text{def}}{=} \{B_1, ..., B_K\}$ denote disjoint sets of propositional atoms. More precisely, $\varphi$ and $\psi$ denote generic propositional formulas built on $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{A} \cup \mathbf{B}$ respectively; $\eta$ and $\mu$ denote total and a partial assignments on $\mathbf{A}$ respectively; $\delta$ denote total assignments on $\mathbf{B}$. (All above symbols may possibly have subscripts.)

*Syntax.* A *propositional formula* is defined inductively as follows: the constants $\top$ and $\bot$ (denoting the truth values true and false) are formulas; a propositional atom $A_1, A_2, A_3, ...$ is a formula; if $\varphi_1$ and $\varphi_2$ are formulas, then $\neg\varphi_1$ and $\varphi_1 \wedge \varphi_2$ are formulas. We use the standard Boolean abbreviations: "$\varphi_1 \vee \varphi_2$" for "$\neg(\neg\varphi_1 \wedge \neg\varphi_2)$", "$\varphi_1 \rightarrow \varphi_2$" for "$\neg(\varphi_1 \wedge \neg\varphi_2)$", "$\varphi_1 \leftrightarrow \varphi_2$" for "$\neg(\varphi_1 \wedge \neg\varphi_2) \wedge \neg(\varphi_2 \wedge \neg\varphi_1)$". A *literal* is either an atom (a *positive literal*) or its negation (a *negative literal*). (If $l$ is a negative literal $\neg A_i$, then by "$\neg l$" we conventionally mean $A_i$ rather than $\neg\neg A_i$.) A

| | | | |
|---|---|---|---|
| $\neg\top$ | $\Rightarrow \bot$ | $\neg\bot$ | $\Rightarrow \top$ |
| $\top \wedge \varphi, \varphi \wedge \top$ | $\Rightarrow \varphi$ | $\bot \wedge \varphi, \varphi \wedge \bot$ | $\Rightarrow \bot$ |
| $\top \vee \varphi, \varphi \vee \top$ | $\Rightarrow \top$ | $\bot \vee \varphi, \varphi \vee \bot$ | $\Rightarrow \varphi$ |
| $\top \rightarrow \varphi$ | $\Rightarrow \varphi$ | $\bot \rightarrow \varphi$ | $\Rightarrow \top$ |
| $\varphi \rightarrow \top$ | $\Rightarrow \top$ | $\varphi \rightarrow \bot$ | $\Rightarrow \neg\varphi$ |
| $\top \leftrightarrow \varphi, \varphi \leftrightarrow \top$ | $\Rightarrow \varphi$ | $\bot \leftrightarrow \varphi, \varphi \leftrightarrow \bot$ | $\Rightarrow \neg\varphi$ |

**Fig. 1.** Propagation of truth values through the Boolean connectives.

| $\mu(\varphi_1)$ | T | T | T | ? | ? | ? | F | F | F |
|---|---|---|---|---|---|---|---|---|---|
| $\mu(\varphi_2)$ | T | ? | F | T | ? | F | T | ? | F |
| $\mu(\neg\varphi_1)$ | F | F | F | ? | ? | ? | T | T | T |
| $\mu(\varphi_1 \wedge \varphi_2)$ | T | ? | F | ? | ? | F | F | F | F |
| $\mu(\varphi_1 \vee \varphi_2)$ | T | T | T | T | ? | ? | T | ? | F |
| $\mu(\varphi_1 \rightarrow \varphi_2)$ | T | ? | F | T | ? | ? | T | T | T |
| $\mu(\varphi_1 \leftrightarrow \varphi_2)$ | T | ? | F | ? | ? | ? | F | ? | T |

**Fig. 2.** Three-value-semantics of $\mu(\varphi)$ in terms of $\{\mathsf{T}, \mathsf{F}, ?\}$.

*clause* is a disjunction of literals $\bigvee_j l_j$. A *cube* is a conjunction of literals $\bigwedge_j l_j$. $\varphi$ is in *Conjunctive Normal Form (CNF)* iff it is a conjunction of clauses: $\bigwedge_{i=1}^{L} \bigvee_{j_i=1}^{K_i} l_{j_i}$.

*Semantics.* Given $\mathbf{A} \stackrel{\text{def}}{=} \{A_1, ..., A_N\}$, a map $\eta : \mathbf{A} \longmapsto \{\mathsf{T}, \mathsf{F}\}$ is a *total truth assignment* for $\mathbf{A}$. We assume $\eta(\top) \stackrel{\text{def}}{=} \mathsf{T}$ and $\eta(\bot) \stackrel{\text{def}}{=} \mathsf{F}$. We represent $\eta$ as a *set of literals* $\eta \stackrel{\text{def}}{=} \{A_i \mid \eta(A_i) = \mathsf{T}\} \cup \{\neg A_i \mid \eta(A_i) = \mathsf{F}\}$. We sometimes represent $\eta$ also as a *cube* $\bigwedge_{\eta(A_i)=\mathsf{T}} A_i \wedge \bigwedge_{\eta(A_i)=\mathsf{F}} \neg A_i$ which we denote as "$\bigwedge\eta$" so that to distinguish the set and the cube representations. We denote by $|\eta|$ the number of literals in $\eta$.

Given a *total* truth assignment $\eta$ on $\mathbf{A}$ and some formulas $\varphi, \varphi_1, \varphi_2$ on $\mathbf{A}$, the sentence "$\eta$ *satisfies* $\varphi$", written "$\eta \models \varphi$", is defined recursively on the structure of $\varphi$ as follows: $\eta \models \top$, $\eta \not\models \bot$, $\eta \models A_i$ if and only if $\eta(A_i) = \mathsf{T}$, $\eta \models \neg\varphi_1$ if and only if $\eta \not\models \varphi_1$, $\eta \models \varphi_1 \wedge \varphi_2$ if and only if $\eta \models \varphi_1$ and $\eta \models \varphi_2$. (The definition of $\eta \models \varphi_1 \bowtie \varphi_2$ for the other connectives follows straightforwardly from their definition in terms of $\neg, \wedge$.) $\varphi$ is *satisfiable* iff $\eta \models \varphi$ for some total truth assignment $\eta$ on $\mathbf{A}$. $\varphi$ is *valid* (written "$\models \varphi$") iff $\eta \models \varphi$ for every total truth assignment $\eta$ on $\mathbf{A}$. $\varphi_1$ *entails* $\varphi_2$ (written "$\varphi_1 \models \varphi_2$") iff, for every total assignment $\eta$ on $\mathbf{A}$, if $\eta \models \varphi_1$ then $\eta \models \varphi_2$. $\varphi_1$ and $\varphi_2$ are *equivalent* (written "$\varphi_1 \equiv \varphi_2$") iff $\varphi_1 \models \varphi_2$ and $\varphi_2 \models \varphi_1$. Consequently: $\varphi$ is unsatisfiable iff $\neg\varphi$ is valid; $\varphi_1 \models \varphi_2$ iff $\varphi_1 \rightarrow \varphi_2$ is valid; a clause $\bigvee_i l_i$ is valid (aka is a *tautology*) iff both $A_i$ and $\neg A_i$ occur in it for some $A_i$; a CNF formula $\varphi$ is valid iff either it is $\top$ or all its clauses are tautologies. We say that a CNF formula is *tautology-free* iff none of its clauses is a tautology.

*Partial truth assignments.* A map $\mu : \mathbf{A}' \longmapsto \{\mathsf{T}, \mathsf{F}\}$ s.t. $\mathbf{A}' \subseteq \mathbf{A}$ and $N' \stackrel{\text{def}}{=} |\mathbf{A}'|$ is a *partial truth assignment* for $\mathbf{A}$. As with total assignments, we can represent $\mu$ as a set of literals or as a cube, denoted with "$\bigwedge\mu$".

By "*apply a partial assignment* $\mu$ *to* $\varphi$" we mean "substitute all instances of each assigned $A_i$ in $\varphi$ with the truth constants in $\{\top, \bot\}$ corresponding to the truth value assigned by $\mu$, and then apply recursively the standard propagation of truth constants through the Boolean connectives described in Figure 1. We denote by "$\varphi|_\mu$" ("*residual of* $\varphi$ *under* $\mu$") the formula resulting from applying $\mu$ to $\varphi$.

We introduce a three-value logic to extend $\mu$ to $\mathbf{A}$ as $\mu : \mathbf{A} \longmapsto \{\mathsf{T}, \mathsf{F}, ?\}$ by assigning to ? (unknown) the unassigned atoms in $\mathbf{A} \setminus \mathbf{A}'$. Then we extend the semantics of $\mu$ to any formula $\varphi$ on $\mathbf{A}$ as described in Figure 2.

The following facts follow straightforwardly and are of interest for our discussion.

**Property 1** *Let $\eta$ be a total truth assignment on $\mathbf{A}$ and $\varphi, \varphi_1, \varphi_2$ be formulas on $\mathbf{A}$.*

(i) *$\eta \models \varphi$ iff $\bigwedge \eta \models \varphi$.*

(ii) *If $\varphi_1$ and $\varphi_2$ are equivalent, then $\eta \models \varphi_1$ iff $\eta \models \varphi_2$.*

(iii) *$\eta \models \varphi$ iff $\varphi|_\eta$ is $\top$.*

(iv) *Checking $\eta \models \varphi$ requires at most a polynomial amount of steps.*

Notice that Property 1(i) justifies the usage of "$\models$" for both satisfiability and entailment.

**Property 2** *Let $\mu$ be a partial assignment on $\mathbf{A}$ and $\varphi$ be a formula on $\mathbf{A}$. Then $\varphi|_\mu$ is $\top$ iff $\mu(\varphi) = \mathsf{T}$ and $\varphi|_\mu$ is $\bot$ iff $\mu(\varphi) = \mathsf{F}$.*

Notice that total assignments are a subcase of partial ones, so that the definition of residual and Property 2 apply also to total assignments $\eta$.

*Existentially-quantified formulas.* A total truth assignment $\eta$ satisfies $\exists\mathbf{B}.\psi$, written "$\eta \models \exists\mathbf{B}.\psi$", iff exists a total truth assignment $\delta$ on $\mathbf{B}$ s.t. $\eta \cup \delta \models \psi$. We call the *Shannon expansion* $\mathsf{SE}[\exists\mathbf{B}.\psi]$ of the existentially-quantified formula $\exists\mathbf{B}.\psi$ the propositional formula on $\mathbf{A}$ defined as

$$\mathsf{SE}[\exists\mathbf{B}.\psi] \stackrel{\text{def}}{=} \bigvee_{\delta_i \in \{\mathsf{T},\mathsf{F}\}^{|\mathbf{B}|}} \psi|_{\delta_i} \tag{1}$$

where $\{\mathsf{T}, \mathsf{F}\}^{|\mathbf{B}|}$ is the set of all total assignments on $\mathbf{B}$. Notice that some $\psi|_{\delta_i}$ may be inconsistent or $\bot$. The following property derives directly from the above definitions.

**Property 3** *Let $\psi$ be a formula on $\mathbf{A} \cup \mathbf{B}$ and $\eta$ be a total truth assignment on $\mathbf{A}$. Then $\eta \models \exists\mathbf{B}.\psi$ iff $\eta \models \mathsf{SE}[\exists\mathbf{B}.\psi]$, that is, $\exists\mathbf{B}.\psi \equiv \mathsf{SE}[\exists\mathbf{B}.\psi]$.*

*CNF-ization.* Every generic formula $\varphi$ on $\mathbf{A}$ can be encoded into a CNF formula $\psi$ on $\mathbf{A} \cup \mathbf{B}$ for some $\mathbf{B}$ by applying (variants of) Tseitin CNF-ization $\mathsf{CNF}_{\mathsf{Ts}}(\varphi)$ [42], consisting in applying recursively bottom-up the rewriting rule:

$$\varphi \Rightarrow \varphi[(l_{j1} \bowtie l_{j2}) \mapsto B_j] \wedge \mathsf{CNF}_{\mathsf{DM}}(B_j \leftrightarrow (l_{j1} \bowtie l_{j2})) \tag{2}$$

until the resulting formula $\psi$ is in CNF, where $l_{j1}, l_{j2}$ are literals, $\bowtie \in \{\wedge, \vee, \rightarrow, \leftarrow, \leftrightarrow\}$ and $\mathsf{CNF}_{\mathsf{DM}}()$ is the validity-preserving CNF conversion based on DeMorgan rules (e.g., $\mathsf{CNF}_{\mathsf{DM}}(B \leftrightarrow (l_1 \wedge l_2)) \stackrel{\text{def}}{=} (\neg B \vee l_1) \wedge (\neg B \vee l_2) \wedge (B \vee \neg l_1 \vee \neg l_2)$). The size of $\psi$ is linear wrt. that of $\varphi$. With Plaisted&Greenbaum CNF-ization $\mathsf{CNF}_{\mathsf{PG}}(\varphi)$ [32] the term $B_j \leftrightarrow (l_{j1} \bowtie l_{j2})$ in (2) is substituted with $B_j \rightarrow (l_{j1} \bowtie l_{j2})$ [resp. $B_j \leftarrow (l_{j1} \bowtie l_{j2})$] if $(l_{j1} \bowtie l_{j2})$ occurs only positively [resp. negatively] in $\varphi$.

**Property 4** *If $\psi \stackrel{\text{def}}{=} \mathsf{CNF}_{\mathsf{Ts}}(\varphi)$ or $\psi \stackrel{\text{def}}{=} \mathsf{CNF}_{\mathsf{PG}}(\varphi)$, then $\eta \models \varphi$ iff exists a total assignment $\delta$ on $\mathbf{B}$ s.t. $\eta \cup \delta \models \psi$, that is, $\varphi \equiv \exists\mathbf{B}.\psi$.*

## 3 A theoretical analysis of verification and entailment

We address the following basic question: *What should we mean by "a partial assignment $\mu$ satisfies $\varphi$"?* We wish to provide a satisfactory definition of partial-assignment satisfiability for a generic propositional formula —i.e., not only for (tautology-free) CNF. Ideally, a suitable definition of partial-assignment satisfiability should verify all statements in Property 1, in particular (ii) and (iv). In practice, unfortunately, at least for generic (i.e., non-CNF) formulas, we show this is not the case.

### 3.1 Verification and entailment of formulas.

*Definitions.* The first candidate definition of partial-assignment satisfaction, which extends to partial assignments Property 1(iii), is *verification*.

**Definition 1.** *We say that a* partial *truth assignment* $\mu$ **verifies** $\varphi$ *iff* $\mu(\varphi) = \mathsf{T}$ *—or, equivalently by Property 2, iff* $\varphi|_\mu = \top$. *We denote this fact with "*$\mu \models\!\!\!\approx \varphi$*".*

We stress the fact that verification is a *semantic* definition ($\mu(\varphi) = \mathsf{T}$) which, due to Property 2, captures exactly the *syntactic* and easy-to-check satisfaction criterion ($\varphi|_\mu = \top$) which is implicitly used in many procedures.

The second candidate definition of partial-assignment satisfaction, which extends to partial assignments Property 1(i), is *entailment*.

**Definition 2.** *We say that a* partial *truth assignment* $\mu$ **entails** $\varphi$ *if and only if, for every total truth assignments* $\eta$ *s.t.*$\mu \subseteq \eta$, $\eta$ *satisfies* $\varphi$ *—or, equivalently, iff* $\varphi|_\mu$ *is valid. We denote this fact with "*$\mu \models \varphi$*".*

*Verification vs. Entailment.* We show the following facts. When the formula $\varphi$ is a tautology-free CNF then verification and entailment coincide: $\mu \models\!\!\!\approx \varphi$ iff $\mu \models \varphi$. Unfortunately, *with generic formulas verification and entailment do not coincide, the former being strictly stronger than the latter.*

**Theorem 1.** *Let* $\varphi$ *be a formula and* $\mu$ *be a partial truth assignment over its atoms.*

*(a) If* $\varphi$ *a tautology-free CNF, then* $\mu \models \varphi$ *iff* $\mu \models\!\!\!\approx \varphi$.
*(b) If* $\varphi$ *is a generic formula, then* $\mu \models \varphi$ *if* $\mu \models\!\!\!\approx \varphi$, *but the converse does not hold in general.*

*Proof.* $(a)$: Let $\varphi$ be a tautology-free CNF. If $\mu \models \varphi$, then $\varphi|_\mu$ is a valid CNF formula which does not contain valid clauses, so that $\varphi|_\mu = \top$, hence $\mu \models\!\!\!\approx \varphi$ by Property 2.
$(a)$ and $(b)$: If $\mu \models\!\!\!\approx \varphi$ then, for every $\eta$ s.t. $\eta \supseteq \mu$, $\eta \models\!\!\!\approx \varphi$ and thus $\eta \models \varphi$, hence $\mu \models \varphi$.
$(b)$: The fact that the converse does not hold in general is shown in Example 1. □

*Example 1.* If $\varphi \stackrel{\text{def}}{=} (A_1 \wedge A_2) \vee (A_1 \wedge \neg A_2)$ and $\mu \stackrel{\text{def}}{=} \{A_1\}$, then $\mu \models \varphi$ but $\mu \not\models\!\!\!\approx \varphi$. ⋄

Hereafter we implicitly assume w.l.o.g. that all CNF formulas are tautology free.

We try to build a counterpart of Property 1 for Definitions 1 and 2 respectively, but in both cases we fail to achieve all points (i)-(iv), resulting into complementary situations. The following properties follow straightforward from Definitions 1 and 2. (Here "✓" [resp. "✗"] denotes facts from Property 1 which are [resp. are not] preserved.)

**Property 5** *Let* $\mu$ *be a partial truth assignment on* $\mathbf{A}$ *and* $\varphi, \varphi_1, \varphi_2$ *be formulas on* $\mathbf{A}$.

(i) ✗ *If* $\mu \models\!\!\!\approx \varphi$ *then* $\bigwedge\mu \models \varphi$, *but not vice versa.*
(ii) ✗ *If* $\varphi_1$ *and* $\varphi_2$ *are equivalent, this does not imply that* $\mu \models\!\!\!\approx \varphi_1$ *iff* $\mu \models\!\!\!\approx \varphi_2$.
(iii) ✓ $\mu \models\!\!\!\approx \varphi$ *iff* $\varphi|_\mu$ *is* $\top$ *(also, iff* $\mu(\varphi) = \mathsf{T}$ *by Property 2).*
(iv) ✓ *Checking* $\mu \models\!\!\!\approx \varphi$ *requires at most a polynomial amount of steps.*

**Property 6** *Let $\mu$ be a partial truth assignment on $\mathbf{A}$ and $\varphi, \varphi_1, \varphi_2$ be formulas on $\mathbf{A}$.*

(i) ✓ $\mu \models \varphi$ *iff* $\bigwedge\mu \models \varphi$.

(ii) ✓ *If $\varphi_1$ and $\varphi_2$ are equivalent, then $\mu \models \varphi_1$ iff $\mu \models \varphi_2$.*

(iii) ✗ $\mu \models \varphi$ *iff* $\varphi|_\mu$ *is a valid formula, not necessarily $\top$ (also, in general $\mu(\varphi) \neq \mathsf{T}$).*

(iv) ✗ *Checking $\mu \models \varphi$ is* co-NP-complete.

*Example 2.* Let $\varphi_1 \stackrel{\text{def}}{=} (A_1 \wedge A_2) \vee (A_1 \wedge \neg A_2)$, $\varphi_2 \stackrel{\text{def}}{=} A_1$ and $\mu \stackrel{\text{def}}{=} \{A_1\}$. Then $\varphi_1 \equiv \varphi_2$, $\varphi_1|_\mu = A_2 \vee \neg A_2 \neq \top$ and $\varphi_2|_\mu = \top$. Thus $\bigwedge\mu \models \varphi_1$, $\mu \models \varphi_1$ and $\mu \models \varphi_2$, whereas $\mu \not\approx \varphi_1$ but $\mu \approx \varphi_2$. ◇

From a theoretical viewpoint, Example 2 spotlights the difference between Property 5(i) and Property 6(i) and, even more importantly, that between Property 5(ii) and Property 6(ii): *whereas entailment matches the intuition that equivalent formulas should be satisfied by the same (partial) assignments, verification does not*. The latter fact looks theoretically awkward. In particular, if we adopted verification as the definition of partial-assignment satisfiability, then we believe that it would be embarrassing to have that equivalent formulas were not "satisfied" by the same assignments.

### 3.2 Verification and entailment of existentially-quantified formulas.

We extend the analysis of §3.1 to existentially-quantified propositional formulas, wishing to provide a satisfactory definition of partial-assignment satisfiability for this case as well. Property 3 paves our way, telling how to extend the definitions of verification and entailment to existentially-quantified formulas.

The first possibility is to see partial-assignment satisfiability as *verification*, leveraging Definition 1 to the existentially-quantified case by exploiting Shannon expansion.

**Definition 3.** *We say that a* partial *truth assignment $\mu$ on $\mathbf{A}$* **verifies** *$\exists\mathbf{B}.\psi$ if and only if, there exists a total truth assignment $\delta$ on $\mathbf{B}$ s.t. $\mu \cup \delta \approx \psi$, that is, $\psi|_{\mu\cup\delta} = \top$.*

**Theorem 2.** *Let $\psi$ be a formula on $\mathbf{A} \cup \mathbf{B}$ and $\mu$ be a partial assignment on $\mathbf{A}$. Then $\mu \approx \exists\mathbf{B}.\psi$ iff $\mu \approx \mathsf{SE}[\exists\mathbf{B}.\psi]$.*

*Proof.* By Definition 1, $\mu \approx \mathsf{SE}[\exists\mathbf{B}.\psi]$ iff $(\mathsf{SE}[\exists\mathbf{B}.\psi])|_\mu = \top$, that is, by (1), iff there exists some $\delta_i$ s.t. $\psi|_{\delta_i}|_\mu = \top$ (i.e., $\psi|_{\delta_i\cup\mu} = \top$), that is, by Definition 1, iff there exists some $\delta_i$ s.t. $\mu \cup \delta_i \approx \psi$, that is, by Definition 3, iff $\mu \approx \exists\mathbf{B}.\psi$. □

Notice that Theorem 2 is *not* a straighforward consequence of Property 3, because equivalent formulas are not necessarily verified by the same assignments (Property 5(ii)).

One second possibility is to see partial-assignment satisfiability as *entailment*, leveraging Definition 2 and Property 3 to the existentially-quantified case. This leads to the following definition and relative Property.

**Definition 4.** *We say that a* partial *truth assignment $\mu$ on $\mathbf{A}$* **entails** *$\exists\mathbf{B}.\psi$, written $\mu \models \exists\mathbf{B}.\psi$, if and only if , for every total truth assignment $\eta$ on $\mathbf{A}$ extending $\mu$, there exists a total truth assignment $\delta$ on $\mathbf{B}$ s.t. $\eta \cup \delta$ satisfies $\psi$.*

**Theorem 3.** *Let $\psi$ be a formula on $\mathbf{A} \cup \mathbf{B}$ and $\mu$ be a partial assignment on $\mathbf{A}$. Then $\mu \models \exists\mathbf{B}.\psi$ iff $\mu \models \mathsf{SE}[\exists\mathbf{B}.\psi]$.*

*Proof.* $\mu \models \mathsf{SE}[\exists\mathbf{B}.\psi]$ iff $\eta \models \mathsf{SE}[\exists\mathbf{B}.\psi]$ for every total assignment $\eta$ s.t. $\eta \supseteq \mu$, that is, by Property 3, iff for every total assignments $\eta$ s.t. $\eta \supseteq \mu$ exists a total assignment $\delta$ on $\mathbf{B}$ s.t. $\eta \cup \delta \models \psi$, that is, iff $\mu \models \exists\mathbf{B}.\psi$. □

Notice the nesting order of forall/exists in Definition 4: "for every $\eta$ exists $\delta$ s.t. ...". In fact, distinct $\eta$'s may satisfy distinct disjuncts $\psi|_{\delta_i}$ in $\mathsf{SE}[\exists\mathbf{B}.\psi]$, requiring distinct $\delta_i$'s.

**Theorem 4.** *Let $\psi$ be a formula on $\mathbf{A} \cup \mathbf{B}$ and $\mu$ be a partial truth assignment over $\mathbf{A}$. Then $\mu \models \exists\mathbf{B}.\psi$ if $\mu \models\!\!\approx \exists\mathbf{B}.\psi$, but the converse does not hold in general.*

*Proof.* From Theorem 1$(b)$ with $\varphi \stackrel{\text{def}}{=} \mathsf{SE}[\exists\mathbf{B}.\psi]$ we can deduce that $\mu \models \mathsf{SE}[\exists\mathbf{B}.\psi]$ if $\mu \models\!\!\approx \mathsf{SE}[\exists\mathbf{B}.\psi]$. By Theorem 2 and Theorem 3 we have that $\mu \models \exists\mathbf{B}.\psi$ if $\mu \models\!\!\approx \exists\mathbf{B}.\psi$. The fact that the converse does not hold in general is shown in Example 3. □

*Example 3.* Consider $\mu \stackrel{\text{def}}{=} \{A_1\}$ and the tautology-free CNF formula on $\mathbf{A} \cup \mathbf{B}$:

$$\begin{aligned}\psi \stackrel{\text{def}}{=}\ & (B_1 \vee B_2) \wedge \\ & (\neg B_1 \vee\ \ A_1) \wedge (\neg B_1 \vee\ \ A_2) \wedge (B_1 \vee \neg A_1 \vee \neg A_2) \wedge \\ & (\neg B_2 \vee\ \ A_1) \wedge (\neg B_2 \vee \neg A_2) \wedge (B_2 \vee \neg A_1 \vee\ \ A_2).\end{aligned}$$

$\mathsf{SE}[\exists\mathbf{B}.\psi] = \cancel{(A_1 \wedge A_2 \wedge \neg A_2)} \vee (A_1 \wedge A_2) \vee (A_1 \wedge \neg A_2) \vee \cancel{\bot}$.

Thus $\mu \models \mathsf{SE}[\exists\mathbf{B}.\psi]$ but $\mu \not\models\!\!\approx \mathsf{SE}[\exists\mathbf{B}.\psi]$, so that $\mu \models \exists\mathbf{B}.\psi$ but $\mu \not\models\!\!\approx \exists\mathbf{B}.\psi$. ◇

Thus verification is strictly stronger than entailment also with existentially-quantified formulas. Remarkably, and unlike with the un-quantified case, this is the case *even if $\psi$ is a tautology-free CNF formula*! (See Example 3.) Intuitively, this can be seen as a consequence of the fact that $\mathsf{SE}[\exists\mathbf{B}.\psi]$ is not in CNF even if $\psi$ is in CNF.

### 3.3 Verification and entailment of CNF-ized non-CNF formulas.

One might argue that in SAT-related problems the distinction between verification and entailment in §3.1 is not relevant in practice, because we could CNF-ize upfront the input non-CNF formulas and then remove tautological clauses, exploiting Property 4 and the fact that the above distinction does not hold for (tautology-free) CNF formulas.

Unfortunately, the following result shows this is not the case.

**Theorem 5.** *Let $\varphi$ be a non-CNF formula on $\mathbf{A}$ and $\psi \stackrel{\text{def}}{=} \mathsf{CNF_{Ts}}(\varphi)$ or $\psi \stackrel{\text{def}}{=} \mathsf{CNF_{PG}}(\varphi)$, and let $\mu$ be a partial truth assignment over $\mathbf{A}$. Then $\mu \models \exists\mathbf{B}.\psi$ if $\mu \models\!\!\approx \exists\mathbf{B}.\psi$, but the converse does not hold in general.*

*Proof.* As an instantiation of Theorem 4 we have $\mu \models \exists\mathbf{B}.\psi$ if $\mu \models\!\!\approx \exists\mathbf{B}.\psi$. The fact that the converse does not hold in general is shown in Example 4. □

*Example 4.* Consider $\varphi \stackrel{\text{def}}{=} (A_1 \wedge A_2) \vee (A_1 \wedge \neg A_2)$ and $\mu \stackrel{\text{def}}{=} \{A_1\}$. Then $\psi \stackrel{\text{def}}{=} \mathsf{CNF_{Ts}}(\varphi)$ is the formula in Example 3. Then $\mu \models \exists\mathbf{B}.\mathsf{CNF_{Ts}}(\varphi)$ but $\mu \not\models\!\!\approx \exists\mathbf{B}.\mathsf{CNF_{Ts}}(\varphi)$.

Let $\psi \stackrel{\text{def}}{=} \mathsf{CNF_{PG}}(\varphi)$ instead. Since $\psi$ is $\mathsf{CNF_{Ts}}(\varphi)$ minus its two ternary clauses, we also have that $\mathsf{SE}[\exists\mathbf{B}.\psi] = \cancel{(A_1 \wedge A_2 \wedge \neg A_2)} \vee (A_1 \wedge A_2) \vee (A_1 \wedge \neg A_2) \vee \cancel{\bot}$.

Thus $\mu \models \exists\mathbf{B}.\mathsf{CNF_{PG}}(\varphi)$ but $\mu \not\models\!\!\approx \exists\mathbf{B}.\mathsf{CNF_{PG}}(\varphi)$. ◇

Intuitively, the whole point is that both $\mathsf{CNF}_{\mathsf{Ts}}(\varphi)$ and $\mathsf{CNF}_{\mathsf{PG}}(\varphi)$ introduce fresh variables $\mathbf{B}$, which must be implicitly existentially quantified away in order to preserve equivalence and thus produce the set of original satisfying assignments (Property 4). Although these variables do not affect entailment, they may affect verification. In fact, since $\varphi$ and $\exists\mathbf{B}.\mathsf{CNF}_{\mathsf{Ts}}(\varphi)$ [resp. $\exists\mathbf{B}.\mathsf{CNF}_{\mathsf{PG}}(\varphi)$] are equivalent, they are entailed by the same partial assignments, but they are not verified by the same partial assignments.

To this extent, in [25] we addressed a different though related problem: we proved that —unlike with $\psi \stackrel{\text{def}}{=} \mathsf{CNF}_{\mathsf{Ts}}(\varphi)$ or with $\psi \stackrel{\text{def}}{=} \mathsf{CNF}_{\mathsf{PG}}(\varphi)$— if $\psi \stackrel{\text{def}}{=} \mathsf{CNF}_{\mathsf{PG}}(\mathsf{NNF}(\varphi))$ and $\mu \not\approx \varphi$, then $\mu \not\approx \exists\mathbf{B}.\psi$. Notice, however, that this fact does not apply to our problem, because Theorem 5 holds also for $\psi \stackrel{\text{def}}{=} \mathsf{CNF}_{\mathsf{PG}}(\mathsf{NNF}(\varphi))$.

*Example 5.* Consider $\varphi$ and $\mu$ as in Example 4. Since $\varphi$ is already in NNF, then we have that $\psi \stackrel{\text{def}}{=} \mathsf{CNF}_{\mathsf{PG}}(\mathsf{NNF}(\varphi)) = \mathsf{CNF}_{\mathsf{PG}}(\varphi)$. Therefore $\mu \models \exists\mathbf{B}.\mathsf{CNF}_{\mathsf{PG}}(\mathsf{NNF}(\varphi))$ but $\mu \not\approx \exists\mathbf{B}.\mathsf{CNF}_{\mathsf{PG}}(\mathsf{NNF}(\varphi))$. $\diamond$

### 3.4 Other candidate forms of partial-assignment satisfaction.

Given the above facts, we wonder if we could use other notions of partial-assignment satisfiability. (E.g., we could enrich $\not\approx$ with some form of formula simplification.) In particular, we wonder if there exists some suitable notion of partial-assignment satisfaction which is strictly stronger then entailment and fixes the "embarrassing fact" for verification of Property 5(ii). The following theorem states this is not the case.

**Theorem 6.** *Let $\trianglerighteq$ denote some relation such that*

*(a) for every $\mu$ and $\varphi$, $\mu \models \varphi$ if $\mu \trianglerighteq \varphi$;*
*(b) for every $\eta$ and $\varphi$, if $\eta$ is total for the set of atoms in $\varphi$, then $\eta \trianglerighteq \varphi$ iff $\eta \models \varphi$;*
*(c) for every $\mu$, $\varphi_1$ and $\varphi_2$, if $\mu \trianglerighteq \varphi_1 \wedge \varphi_2$, then $\mu \trianglerighteq \varphi_1$ and $\mu \trianglerighteq \varphi_2$;*
*(d) for every $\mu$, $\varphi_1$ and $\varphi_2$, if $\varphi_1 \equiv \varphi_2$, then $\mu \trianglerighteq \varphi_1$ iff $\mu \trianglerighteq \varphi_2$*

*Then, for every $\mu$, $\varphi$, we have that $\mu \trianglerighteq \varphi$ if and only if $\mu \models \varphi$.*

*Proof.* The "only if" part is hypothesis *(a)*.
The "if" part: By absurd, suppose there exist $\mu$ and $\varphi$ s.t. $\mu \models \varphi$ and $\mu \not\trianglerighteq \varphi$ .
Let $\varphi_1 \stackrel{\text{def}}{=} \bigwedge\mu$ and $\varphi_2 \stackrel{\text{def}}{=} \bigwedge\mu \wedge \varphi$. Then we have that:
$\varphi_1 \equiv \varphi_2$, because $\mu \models \varphi$, and hence $\bigwedge\mu \models \varphi$ because of Property 6(i);
$\mu \trianglerighteq \varphi_1$, because of hypothesis *(b)*;
$\mu \not\trianglerighteq \varphi_2$, because of hypothesis *(c)*.
The latter three facts violate hypothesis *(d)*. $\square$

Theorem 6 says that entailment is the only relation which $(a)$ is stronger or equal than entailment, $(b)$ extends to partial assignments the standard notion of total-assignment satisfaction, $(c)$ maintains the standard semantics of $\wedge$, and $(d)$ verifies Property 6(ii). Thus, any such relation $\trianglerighteq$ which is strictly stronger that $\models$ suffers for the same "embarrassing fact" of Property 5(ii).

*Example 6.* Let "$\trianglerighteq$" denote a variant of $\approx$ such that $\mu \trianglerighteq \varphi$ implements a syntactic check that adds "$l \vee \neg l \Rightarrow \top$", $l$ being a literal, to the rules used in Fig. 1 for computing the residual $\varphi|_\mu$ . Thus, considering the formula $\varphi_1$ of Example 2 we have $\mu \trianglerighteq \varphi_1$.

Nevertheless, let $\mu \stackrel{\text{def}}{=} \{A_1, .., A_M\}$ s.t. $\varphi \stackrel{\text{def}}{=} \bigvee_{i=1}^{M}(A_i \wedge cube_i)$ s.t. each $cube_i$ is a non-unary cube and $\bigvee_{i=1}^{M} cube_i$ is valid and does not contain occurrences of the atoms $A_1, .., A_M$. Then $\mu \models \varphi$ but $\mu \not\trianglerighteq \varphi$, because $\varphi|_\mu$ is the valid formula $\bigvee_i cube_i$ which is not simplified by the above reduction. If $\varphi_2 \stackrel{\text{def}}{=} \bigwedge \mu$, then $\varphi \equiv \varphi_2$, $\mu \not\trianglerighteq \varphi$ but $\mu \trianglerighteq \varphi_2$, whereas $\mu \models \varphi$ and $\mu \models \varphi_2$. Thus the "embarrassing fact" above applies also to $\trianglerighteq$. $\diamond$

# 4 Practical consequences of using verification or entailment

In this section we analyze the practical consequences of adopting verification or entailment as a partial-assignment satisfiability tests in search procedures for satisfiability and enumeration and in formula compilation. In particular, when entailment and verification do not coincide, we analyze the tradeoff between the extra cost of entailment versus the benefits it may produce.

## 4.1 Verification vs. entailment in solving, enumeration and compilation

*Verification vs. entailment for native CNF formulas.* When the input formula is natively in CNF, the distinction between verification and entailment is not relevant, because these two concepts coincide. (We conjecture that this is the main reason why the distinction between verification and entailment has been long overlooked in the literature.) For satisfiability of CNF formulas, CDCL SAT solvers directly produce *total* truth assignments, because it is not worth to perform intermediate satisfiability checks.

*Verification vs. entailment in satisfiability.* In satisfiability we need finding only one total assignment $\eta$ extending $\mu$ which satisfy $\varphi$. Therefore entailment produces no benefits wrt. verification and is more expensive, so that the latter is always preferred.

The scenario changes when we deal with enumeration-based algorithms applied to non-CNF formulas, or to existentially-quantified formulas, or to CNF-ized formulas.

*Verification vs. entailment for enumeration on non-CNF formulas.* The analysis in §3.1 shows the following facts. On the one hand, the advantage of adopting verification rather than entailment for checking partial-assignment satisfiability is that it matches the intuition and practical need that satisfiability checking should be fast, since checking verification is polynomial (Property 5(iv)) and easier to implement, whereas checking entailment is co-NP-complete (Property 6(iv)), because it is equivalent to checking the validity of the residual $\varphi|_\mu$ . We stress the fact, however, that if $\mu \models \varphi$, then the residual $\varphi|_\mu$ is typically much smaller than $\varphi$ with a much smaller search space, since its variables are only (a subset of) the variables which are not assigned by $\mu$, so that in many actual situations the above fact could not be a major issue.

On the other hand, a main advantage of adopting entailment on enumeration is that, due to Theorem 1, every partial assignments entailing the input formula is a *sub-assignment* of some other(s) verifying it. Consequently, as soon as one verification-based enumeration procedure produces (a branch corresponding to) an assignment $\mu$

s.t. $\mu \models \varphi$ but $\mu \not\approx \varphi$, it cannot realize this fact and thus proceeds the search to extend $\mu$ to some $\mu' \supset \mu$ s.t. $\mu' \not\approx \varphi$. This causes an extension of the search tree of up to $2^{|\mu'|-|\mu|}$ branches. Therefore, for an assignment-enumeration algorithm, being able to enumerate partial assignments entailing the input formula rather than simply verifying it may (even drastically) reduce the number of the satisfying assignments enumerated.

Also, the analysis in §3.4 shows that alternative forms of partial-assignment satisfiability extending verification —e.g., adding further simplifications for the residual $\varphi|_\mu$ like unit-propagation or the removal of basic valid subformulas— cannot fill the gap wrt. entailment, whose theoretical properties are specific.

*Verification vs. entailment for projected enumeration.* The analysis in §3.2 shows that the above considerations for non-CNF formulas extend automatically to procedures for projected enumeration (e.g. projected AllSAT/AllSMT, projected #SAT/#SMT), even on CNF formulas, because projected enumeration consists in enumeration on an existentially-quantified formula $\exists \mathbf{B}.\psi$, $\mathbf{B}$ being the non-relevant atoms to be projected away. The same considerations apply also to other applications like preimage computation in symbolic model checking [5] or predicate abstraction in SW verification [21,24].

*Verification vs. entailment for CNF-ized non-CNF formulas.* The analysis in §3.3 shows that, although verification and entailment coincide for CNF formulas, the problem with non-CNF formulas cannot be fixed by simply CNF-izing a formula upfront [42,32] and running a CNF enumeration procedure based on partial-assignment verification, projecting out the fresh variables introduced by the CNF-ization. In fact, a partial assignment over the original atoms may entail the existentially-quantified CNF-ized formula without verifying it. Also, even adopting the CNF-ization technique $\mathsf{CNF_{PG}}(\mathsf{NNF}(\varphi))$ we proposed in [25], although improving the enumeration process, does not fill the gap between verification and entailment in this case.

*Verification and entailment in formula compilation.* With formula compilers a formula $\varphi$ is encoded into a format which makes assignment enumeration straightforward [13]. d-DNNF formulas are NNF DAGs which are *decomposable* (i.e., all conjunctive subformulas $\varphi_1 \wedge \varphi_2$ are such that $\varphi_1$ and $\varphi_2$ do not share atoms) and *deterministic* (i.e., all disjunctive subformulas $\varphi_1 \vee \varphi_2$ are such that $\varphi_1$ and $\varphi_2$ are mutually inconsistent) [11]. The d-DNNF compilers typically implement a top-down search, implicitly adopting verification ($\varphi|_\mu = \top$) as partial-assignment satisfiability test [10].

OBDDs [4] and SDDs [12] are subcases of d-DNNFs [13] which are canonical under some order condition —i.e., two equivalent subformulas $\varphi_1$ and $\varphi_2$ are encoded into the same OBDD or SDD, and as such are shared inside the DAG representation. The OBDD and SDD compilers typically build the encoding bottom-up, and they are able to encode $\varphi|_\mu$ into $\top$ as soon as $\varphi|_\mu$ becomes valid. (That is, they implicitly use entailment as partial-assignment satisfiability test.)

*Verification and entailment for #SMT and WMI.* With some enumeration-based techniques (e.g. #SMT [7,31] and WMI [29,39]) the total cost of *processing* each enumerated partial assignment dominates that of the enumeration itself. (E.g., with WMI each satisfying assignment $\mu$ corresponds to a polytope, and for each $\mu$ it is necessary to compute the integral of some multivariate function over such polytope.) Therefore, in this context the *effectiveness* of enumeration in reducing the number of assignments generated is even more important that the *efficiency* in terms of CPU time required to

enumerate them. Therefore, with entailment-based enumeration the reduced number of assignments produced, and hence of integrals computed, should definitely compensate the extra cost of entailment checking.

### 4.2 Verification vs. entailment in search procedures and formula compilers.

When applied to satisfiable formulas, algorithms like *Analytic Tableaux* [3] [38] or "classic" *DPLL* [4] [14], or even recent sophisticated AllSAT procedures for circuits [17], produce branches representing partial assignments which *verify* the input formula. The same happens with enumeration algorithms for d-DNNF formulas, which resemble the tableaux enumeration algorithm, with the restriction that the formulas they are applied to are deterministic and decomposable NNFs, so that the enumerated assignment are pairwise disjoint and straighforward to compute, because the enumeration reduces to the traversal of the and-or DAG without encountering inconsistent branches [10].

*Example 7.* Consider $\varphi \stackrel{\text{def}}{=} ((A_1 \wedge A_2) \vee (A_1 \wedge \neg A_2)) \wedge ((\neg A_3 \wedge A_4) \vee (\neg A_3 \wedge \neg A_4))$. Notice that the single assignment $\mu \stackrel{\text{def}}{=} \{A_1, \neg A_3\}$ subsumes all total truth assignments satisfying $\varphi$ and is such that $\mu \models \varphi$ but $\mu \not\approx \varphi$, since $\varphi|_\mu = (A_2 \vee \neg A_2) \wedge (A_4 \vee \neg A_4)$.

Figure 3 represents the search trees corresponding to AllSAT executions of Analytic Tableaux and (non-CNF) DPLL on $\varphi$[5] [6]. Since (the DAG version of) $\varphi$ is already in d-DNNF (Fig. 4), in this case the d-DNNF enumeration resembles the tableaux enumeration. In all cases, the enumeration produces:
$\{\{A_1, A_2, \neg A_3, A_4\}, \{A_1, A_2, \neg A_3, \neg A_4\}, \{A_1, \neg A_2, \neg A_3, A_4\}, \{A_1, \neg A_2, \neg A_3, \neg A_4\}\}$.
Notice that all the assignments produced are total. Notice also that neither Analytic Tableaux nor DPLL nor d-DNNF enumeration in this case can produce $\{\{A_1, \neg A_3\}\}$, regardless the adopted strategy. The same applies also to the verification-based AllSAT circuit enumeration of [17], see Example 9. $\diamond$

OBDDs [4] contain branches representing partial assignments which *entail* the input formula, because if $\mu \models \varphi$ then $\varphi|_\mu$ is valid (Property 6(iii)), so that its corresponding sub-OBDD is reduced into the $\top$ node. (SDDs [12] behave similarly.)

*Example 8.* Consider $\varphi$ as in Example 7. Figure 4 represents possible compilations of $\varphi$ into d-DNNF, OBDD and SDD respectively. [7] Notice that, unlike with OBDDs and SDDs which collapse equivalent subformulas into one, the two $\vee$-subformulas in the d-DNNF are not recognized to be equivalent to the nodes $A_1$ and $\neg A_3$ respectively.

Applying enumeration to the d-DNNF we obtain the total assignments of Example 7. By applying enumeration to the OBDD or to the SDD, we obtain the single-assignment $\{\{A_1, \neg A_3\}\}$, so that $\{A_1, \neg A_3\} \models \varphi$ but $\{A_1, \neg A_3\} \not\approx \varphi$. $\diamond$

[3] Notice that Analytic Tableaux may generate duplicated or subsumed assignments (see [9,19])

[4] Classic DPLL procedure [14] was designed to work for CNF formulas. Nevertheless it is easy to produce a non-CNF version of this procedure (see e.g. [1]).

[5] Here in DPLL the pure-literal rule [14] is not used because in All-SAT it may hinder the enumeration of relevant models (see, e.g., [33]).

[6] The search trees may vary with the ordering by which variables and subformulas are analyzed.

[7] The outcome may vary with the ordering by which variables and subformulas are analyzed.

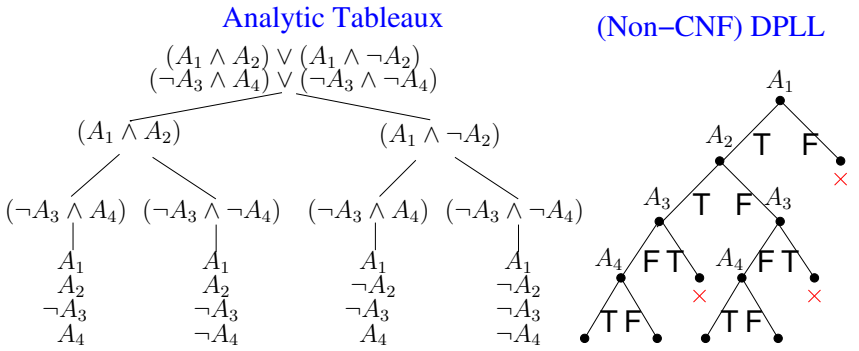


**Fig. 3.** $\varphi \stackrel{\text{def}}{=} ((A_1 \wedge A_2) \vee (A_1 \wedge \neg A_2)) \wedge ((\neg A_3 \wedge A_4) \vee (\neg A_3 \wedge \neg A_4))$.
Enumeration by analytic tableaux (left) and (non-CNF) DPLL (right).

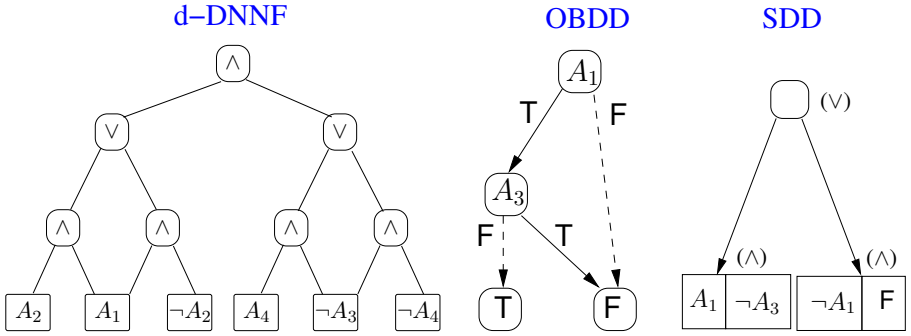


**Fig. 4.** $\varphi \stackrel{\text{def}}{=} ((A_1 \wedge A_2) \vee (A_1 \wedge \neg A_2)) \wedge ((\neg A_3 \wedge A_4) \vee (\neg A_3 \wedge \neg A_4))$.
Compilation of $\varphi$ into: d-DNNF (left), OBDD (center), SDD (right).

### 4.3 Implementing entailment within enumeration procedures

Implementing efficiently entailment-based enumeration procedures can be tricky. One possible approach is to integrate AllSAT/AllSMT with OBDDs, exploiting the capability of OBDD to implicitly perform entailment-based reductions [6]. The main problem with this approach is the fact that OBDDs quickly blow up in memory as soon as the input formula increases in size.

[15,27] proposed a formal reasoning framework based on the idea of *dualized search*: while the main enumerator produces partial assignments $\mu$ for the input formula $\varphi$, another "dual" SAT solver is incrementally called on $\mathsf{CNF}_{\mathsf{Ts}}(\neg\varphi) \wedge \bigwedge\mu$: if it is unsatisfiable then the main enumerator produces $\mu$ and backtracks. Based on the analysis in [35] and noticing that the above dual check in [15,27] is an entailment ($\mu \models \varphi$) rather than a verification ($\mu \not\approx \varphi$), in [28,30] we proposed enumeration and model counting frameworks and algorithms based on dual search, enhanced with chronological backtracking.

Unfortunately no implementation efficient enough to compete with state-of-the-art enumeration procedures was produced.

[17] presented HALL, a sophisticated AllSAT procedure for circuits which, among other techniques, exploited a form of (verification-based) *generalization* (aka minimization or reduction): as soon as a total truth assignment $\eta$ satisfying $\varphi$ is produced, the literals in $\eta$ are removed one by one and the resulting partial assignment is tested to verify $\varphi$. In [16] we have enhanced HALL generalization procedure by substituting verification checks with entailment checks based on dualized search, boosting its performance in terms of both time efficiency and size of partial-assignment sets.

*Example 9.* Consider $\varphi \stackrel{\text{def}}{=} ((A_1 \wedge A_2) \vee (A_1 \wedge \neg A_2)) \wedge ((\neg A_3 \wedge A_4) \vee (\neg A_3 \wedge \neg A_4))$ as in Example 7 and assume some total assignment satisfying it is produced, e.g. $\eta \stackrel{\text{def}}{=} \{A_1, A_2, \neg A_3, A_4\}$. Since there is no literal $l \in \eta$ s.t. $\eta \backslash \{l\} \models \varphi$, no assignment reduction based on verification can be successfully applied to $\eta$. Consequently, even the sophisticated verification-based circuit AllSAT procedure in [17] produces the set of four un-reduced total assignments of Example 7.

Instead, since $\eta \backslash \{A_2, A_4\} \models \varphi$ whereas $\eta \backslash \{A_1\} \not\models \varphi$ and $\eta \backslash \{\neg A_3\} \not\models \varphi$, the entailment-based assignment-reduction technique of [16] shrinks $\eta$ into $\{A_1, \neg A_3\}$, so that the enumeration procedure returns $\{\{A_1, \neg A_3\}\}$. $\diamond$

## 5 Conclusions and Future Work

In this paper we have analyzed in depth the issue of formula satisfaction by partial assignments. We have identified two alternative notions that are implicitly used in the literature, namely verification and entailment, and we have showed that, although the former is easier to check and as such is implicitly used by most current search procedures, the latter has better theoretical properties, and can improve the efficiency and effectiveness of enumeration procedures.

From a theoretical viewpoint, we highlight the need for a unique universally-agreed definition of partial-assignment satisfaction[8], and we champion the idea that it should be defined as entailment, considering verification an easy-to-check sufficient condition for it. From a practical viewpoint, we suggest that entailment should be adopted as partial-assignment satisfaction check inside enumeration-based procedures in place of verification, so that to reduce the number of enumerated assignments.

The analysis presented in this paper opens several research avenues. First, we wish to investigate and evaluate empirically novel entailment-based techniques for both (projected) enumeration and (projected) model counting. In particular, we wish to enhance our (projected) AllSAT procedure in [40,41] with entailment tests. Second, we wish to implement entailment-based enumeration techniques to enhance our AllSMT procedures in MATHSAT [8] and evaluate them empirically. In particular, we wish to embed and evaluate such entailment-based AllSMT procedure inside our WMI tools [29,39]. Third, we wish to embed novel entailment-based AllSMT procedures inside our SMT compilers for $\mathcal{T}$-OBDDs and $\mathcal{T}$-SDDs [26], extending it also to generic $\mathcal{T}$-d-DNNFs.

[8] In 2020 we ran an online poll among SAT and SMT PC members about the meaning of partial-assignment satisfiability: 35% said "entailment", 35% said "verification", 30% said both (!).

## Acknowledgements

The analysis described in this paper has strongly benefitted from interesting discussions, either in person or via email, with (in alphabetic order): Armin Biere, Alessandro Cimatti, Dror Fried, Allen van Gelder (R.I.P.), Alberto Griggio, Gabriele Masina, David Mitchell, Sibylle Möhle, Paolo Morettin, Andrea Passerini, Alexander Nadel, Yogev Shalmon, Laurent Simon, Giuseppe Spallitta, Armando Tacchella, Stefano Tonetta, and Moshe Vardi, whom are all warmly thanked.

The author was supported in part by the MUR PNRR project FAIR - Future AI Research (PE00000013) funded by the NextGenerationEU; he was partially funded under the NRRP, Mission 4 Component 2 Investment 1.4, by the European Union – NextGenerationEU (proj. nr. CN 00000013); he was supported in part by the TANGO project funded by the EU Horizon Europe research and innovation program under GA No 101120763. Views and opinions expressed are however those of the author only and do not necessarily reflect those of the European Union, the European Health and Digital Executive Agency (HaDEA) or The European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

### Disclosure of Interests

The author has no competing interests to declare that are relevant to the content of this article.

## References

1. A. Armando and E. Giunchiglia. Embedding Complex Decision Procedures inside an Interactive Theorem Prover. *Annals of Mathematics and Artificial Intelligence*, 8(3–4):475–502, 1993.
2. R. A. Aziz, G. Chu, C. Muise, and P. Stuckey. #∃SAT: Projected Model Counting. In *Theory and Applications of Satisfiability Testing – SAT 2015*, pages 121–137. Springer, 2015.
3. C. W. Barrett, R. Sebastiani, S. A. Seshia, and C. Tinelli. Satisfiability modulo theories. In A. Biere, M. Heule, H. van Maaren, and T. Walsh, editors, *Handbook of Satisfiability - Second Edition*, volume 336, pages 1267–1329. IOS Press, 2021.
4. R. E. Bryant. Graph-Based Algorithms for Boolean Function Manipulation. *IEEE Transactions on Computers*, C-35(8):677–691, Aug. 1986.
5. J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and L. J. Hwang. Symbolic Model Checking: $10^{20}$ States and Beyond. *Information and Computation*, 98(2):142–170, 1992.
6. R. Cavada, A. Cimatti, A. Franzén, K. Kalyanasundaram, M. Roveri, and R. K. Shyamasundar. Computing Predicate Abstractions by Integrating BDDs and SMT Solvers. In *FMCAD*, pages 69–76. IEEE Computer Society, 2007.
7. D. Chistikov, R. Dimitrova, and R. Majumdar. Approximate counting in smt and value estimation for probabilistic programs. In *TACAS*, pages 320–334. Springer, 2015.
8. A. Cimatti, A. Griggio, B. J. Schaafsma, and R. Sebastiani. The MathSAT 5 SMT Solver. In *TACAS'13.*, volume 7795 of *LNCS*, pages 95–109. Springer, 2013.
9. M. D'Agostino. Are Tableaux an Improvement on Truth-Tables? *Journal of Logic, Language and Information*, 1:235–252, 1992.
10. A. Darwiche. Decomposable negation normal form. *J. ACM*, 48(4):608–647, July 2001.

11. A. Darwiche. A Compiler for Deterministic, Decomposable Negation Normal Form. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 627–634. American Association for Artificial Intelligence, July 2002.
12. A. Darwiche. SDD: A new canonical representation of propositional knowledge bases. In *22nd International Joint Conference on Artificial Intelligence*, volume 2 of *IJCAI'11*, pages 819–826, Barcelona, Catalonia, Spain, July 2011. AAAI Press.
13. A. Darwiche and P. Marquis. A knowledge compilation map. *Journal of Artificial Intelligence Research*, 17(1):229–264, Sept. 2002.
14. M. Davis, G. Longemann, and D. Loveland. A machine program for theorem proving. *Journal of the ACM*, 5(7), 1962.
15. K. Fazekas, M. Seidl, and A. Biere. A duality-aware calculus for quantified boolean formulas. In *SYNASC*, pages 181–186. IEEE Computer Society, 2016.
16. D. Fried, A. Nadel, R. Sebastiani, and Y. Shalmon. Entailing Generalization Boosts Enumeration. In *27th International Conference on Theory and Applications of Satisfiability Testing*, volume 305 of *LIPIcs*, pages 13:1–13:14. Schloss Dagstuhl – LZI, 2024.
17. D. Fried, A. Nadel, and Y. Shalmon. AllSAT for Combinational Circuits. In *26th International Conference on Theory and Applications of Satisfiability Testing*, volume 271 of *LIPIcs*, pages 9:1–9:18. Schloss Dagstuhl – LZI, 2023.
18. M. Gebser, B. Kaufmann, and T. Schaub. Solution Enumeration for Projected Boolean Search Problems. In *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, pages 71–86. Springer, 2009.
19. F. Giunchiglia and R. Sebastiani. Building decision procedures for modal logics from propositional decision procedures - the case study of modal K(m). *Information and Computation*, 162(1/2), October/November 2000.
20. C. P. Gomes, A. Sabharwal, and B. Selman. Model counting. In A. Biere, M. Heule, H. van Maaren, and T. Walsh, editors, *Handbook of Satisfiability - Second Edition*, volume 336, pages 993–1014. IOS Press, 2021.
21. S. Graf and H. Saïdi. Construction of abstract state graphs with pvs. In *CAV*. Springer, 1997.
22. H. Kleine Büning and O. Kullmann. Minimal unsatisfiability and autarkies. In A. Biere, M. Heule, H. van Maaren, and T. Walsh, editors, *Handbook of Satisfiability - Second Edition*, volume 336, pages 571–633. IOS Press, 2021.
23. J.-M. Lagniez and E. Lonca. Leveraging Decision-DNNF Compilation for Enumerating Disjoint Partial Models. In *21st International Conference on Principles of Knowledge Representation and Reasoning*, Nov. 2024.
24. S. K. Lahiri, R. Nieuwenhuis, and A. Oliveras. SMT Techniques for Fast Predicate Abstraction. In *Computer Aided Verification*, LNCS, pages 424–437. Springer, Aug. 2006.
25. G. Masina, G. Spallitta, and R. Sebastiani. On CNF Conversion for Disjoint SAT Enumeration. In *26th International Conference on Theory and Applications of Satisfiability Testing*, volume 271 of *LIPIcs*, pages 15:1–15:16. Schloss Dagstuhl – LZI, 2023.
26. M. Michelutti, G. Masina, G. Spallitta, and R. Sebastiani. Canonical decision diagrams modulo theories. In *ECAI 2024 - 27th European Conference on Artificial Intelligence*, volume 392, pages 4319–4327. IOS Press, 2024.
27. S. Möhle and A. Biere. Dualizing Projected Model Counting. In *30th IEEE International Conference on Tools with Artificial Intelligence*, pages 702–709, Nov. 2018.
28. S. Möhle, R. Sebastiani, and A. Biere. Four flavors of entailment. *Theory and Applications of Satisfiability Testing – SAT 2020*, 12178:62 – 71, 2020.
29. P. Morettin, A. Passerini, and R. Sebastiani. Advanced SMT techniques for weighted model integration. *Artificial Intelligence*, 275:1–27, October 2019.
30. S. Möhle, R. Sebastiani, and A. Biere. On enumerating short projected models. *Discrete Applied Mathematics*, 361:412–439, 2025.

31. Q. Phan and P. Malacaria. All-solution satisfiability modulo theories: Applications, algorithms and benchmarks. In *ARES*. IEEE Computer Society, 2015.
32. D. A. Plaisted and S. Greenbaum. A Structure-preserving Clause Form Translation. *Journal of Symbolic Computation*, 2(3):293–304, 1986.
33. R. Sebastiani. From KSAT to Delayed Theory Combination: Exploiting DPLL Outside the SAT Domain. In *Frontiers of Combining Systems, 6th International Symposium, FroCoS*, volume 4720 of *LNCS*, pages 28–46. Springer, 2007.
34. R. Sebastiani. Lazy Satisfiability Modulo Theories. *Journal on Satisfiability, Boolean Modeling and Computation, JSAT*, 3(3-4):141–224, 2007.
35. R. Sebastiani. Are You Satisfied by This Partial Assignment? arXiv preprint arXiv:2003.04225, Feb. 2020.
36. R. Sebastiani and A. Tacchella. SAT techniques for modal and description logics. In A. Biere, M. Heule, H. van Maaren, and T. Walsh, editors, *Handbook of Satisfiability - Second Edition*, volume 336, pages 1223–1266. IOS Press, 2021.
37. R. Sebastiani and S. Tomasi. Optimization Modulo Theories with Linear Rational Costs. *ACM Transactions on Computational Logics*, 16(2), March 2015.
38. R. M. Smullyan. *First-Order Logic*. Springer, 1968.
39. G. Spallitta, G. Masina, P. Morettin, A. Passerini, and R. Sebastiani. Enhancing SMT-based Weighted Model Integration by Structure Awareness. *Artificial Intelligence*, 328:104067, Mar. 2024.
40. G. Spallitta, R. Sebastiani, and A. Biere. Disjoint Partial Enumeration without Blocking Clauses. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 8126–8135, Mar. 2024.
41. G. Spallitta, R. Sebastiani, and A. Biere. Disjoint Projected Enumeration for SAT and SMT without Blocking Clauses. *Artificial Intelligence*, 345:104346, April 2025.
42. G. Tseitin. On the complexity of derivation in propositional calculus. In *studies in constructive mathematics and mathematical logics*, pages 115–125, 1970.