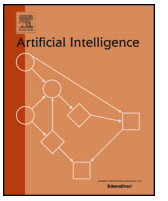
Artificial Intelligence 328 (2024) 104077

Contents lists available at ScienceDirect

Artificial Intelligence

journal homepage: www.elsevier.com/locate/artint



On the role of logical separability in knowledge compilation

Junming Qiu [a,e], Wenqing Li [a], Liangda Fang [a,f,*], Quanlong Guan [a,d,*], Zhanhao Xiao [b], Zhao-Rong Lai [a], Qian Dong [c]

[a] Jinan University, Guangzhou, 510632, Guangdong, China
[b] Guangdong Polytechnic Normal University, Guangzhou, 510665, Guangdong, China
[c] Xi'an Jiaotong-Liverpool University, Suzhou, 215123, Jiangsu, China
[d] Guangdong-Macao Advanced Intelligent Computing Joint Laboratory, Zhuhai, 519031, Guangdong, China
[e] Sun Yat-sen University, Guangzhou, 510006, Guangdong, China
[f] Pazhou Lab, Guangzhou, 510330, Guangdong, China

ARTICLE INFO

Keywords:
Knowledge compilation
Logical separability
Propositional logic

ABSTRACT

Knowledge compilation is an alternative solution to address demanding reasoning tasks with high complexity via converting knowledge bases into a suitable target language. The notion of logical separability, proposed by Levesque, offers a general explanation for the tractability of clausal entailment for two remarkable languages: decomposable negation normal form and prime implicates. It is interesting to explore what role logical separability plays in problem tractability. In this paper, we apply the notion of logical separability to a number of reasoning problems within the context of propositional logic: satisfiability checking (CO), clausal entailment checking (CE), model counting (CT), model enumeration (ME) and forgetting (FO), as well as their dual tasks, contributing to several recursive procedures. We provide the corresponding logical separability based properties: CO-logical separability, CE-logical separability, CT-logical separability, ME-logical separability and their duals. Based on these properties, we then identify four novel normal forms: CO-LSNNF, CE-LSNNF, CT-LSNNF and ME-LSNNF, as well as their dual languages. We show that each of them is the necessary and sufficient condition under which the corresponding procedure is correct. We finally integrate the above normal forms into the knowledge compilation map.

1. Introduction

Knowledge compilation (KC) has been attracted interests in many areas of AI, such as, model-based diagnosis [1,2], explainable machine learning [3,4], probabilistic inference [5,6] and planning [7,8] and so on. KC splits the reasoning process for demanding reasoning tasks into two phases: an off-line compilation phase and an on-line reasoning phase. In the off-line phase, the knowledge base is first converted into a surrogate in form of a suitable target compilation language, in which the demanding reasoning tasks can be tractably accomplished. Then, the compiled knowledge base can be used to efficiently answer multiple queries, preforming transformations during the on-line phase.

Darwiche and Marquis [9] proposed two criteria to evaluate target compilation languages: succinctness (the size of complied knowledge bases) and tractability (the supported polytime queries and transformations). In general, these two criteria are comple-

* Corresponding authors at: Jinan University, Guangzhou, 510632, Guangdong, China.
E-mail addresses: 2040697476jnu@stu2020.jnu.edu.cn (J. Qiu), eskii0706@stu2020.jnu.edu.cn (W. Li), fangld@jnu.edu.cn (L. Fang), gql@jnu.edu.cn (Q. Guan), xiaozhanhao@gpnu.edu.cn (Z. Xiao), laizhr@jnu.edu.cn (Z.-R. Lai), qian.dong@xjtlu.edu.cn (Q. Dong).

https://doi.org/10.1016/j.artint.2024.104077
Received 9 February 2023; Received in revised form 31 October 2023; Accepted 8 January 2024

Available online 12 January 2024

mentary: more succinct languages are less tractable and vice versa. One of the primary purposes of KC is to design a suitable target language that achieves a good trade-off between succinctness and tractability for one or more specific reasoning tasks. In addition, Darwiche and Marquis studied a dozen proportional languages, and provided a knowledge compilation map that integrates them in terms of their succinctness and tractability. In the past decades, a number of target compilation languages, including OBDD [10], SDD [11], ToOBDD$_<$ [12] and so on, were developed and integrated into a knowledge compilation map. One of the most prominent target languages among them is decomposable negation normal form (DNNF), a subset of negation normal form (NNF) satisfying the $\wedge$-decomposability property [13]. It subsumes most of the existing target languages. Darwiche designed two polytime procedures for satisfiability checking and clausal entailment checking, which are correct for DNNF.

What is the intrinsic reason why DNNF supports such polytime queries? This question can be answered by the notion of *logical separability*, which was first proposed by Levesque [14]. Roughly speaking, a conjunction ϕ is logically separable, if the clausal entailment problem for ϕ can be decomposed to the entailment problems for every conjunct of ϕ. It is easily verified that any conjunction appearing in a DNNF-formula is logically separable due to the $\wedge$-decomposability property. In addition, prime implicate normal form (PI) supports polytime clausal entailment checking as the conjunction of prime implicates is logically separable. Hence, logical separability offers a thorough explanation for why both DNNF and PI supports the clausal entailment checking.

It is interesting to explore what role logical separability plays in problem tractability. In [14], Levesque also defined a normal form, namely Levesque's normal form (LNF), where entailment checking is tractable for a class of knowledge bases. Unfortunately, after that, the idea of logical separability has hardly been applied to language design for other reasoning problems. It motivates us to take logical separability into account to design succinct target languages in which reasoning problems are tractably solved.

In this paper, we focus on several classic reasoning problems in the context of propositional logic, including four queries: satisfiability checking, clausal entailment checking, model counting and model enumeration, as well as the transformation named forgetting. For the four query tasks, we start by providing recursive procedures, $\mathcal{CO}$, $\mathcal{CE}$, $\mathcal{CT}$ and $\mathcal{ME}$, and then give definitions of logical separability based properties: CO-logical separability, CE-logical separability, CT-logical separability and ME-logical separability, respectively. We also propose several novel normal forms: CO-LSNNF, CE-LSNNF, CT-LSNNF and ME-LSNNF, each of which satisfies the above logical separability based properties, respectively, and show that every normal form is the necessary and sufficient condition under which the corresponding procedure is correct. In addition, all of them are complete languages, i.e. for every propositional formula, there exists an equivalent one in any of these logical separability-based normal forms. As for the transformation of forgetting, we show that CE-logical separability is exactly the sufficient and necessary condition to guarantee the correctness of the polytime procedure $\mathcal{FO}$ for forgetting task.

- For CO-LSNNF, we provide two interesting theoretical results: (1) any language supports polytime satisfiability checking iff it is polynomially translatable into CO-LSNNF; and (2) CO-LSNNF is equally succinct to NNF.
- For CE-LSNNF, we show that if a language is polynomially translatable into CE-LSNNF, then it supports polytime clausal entailment checking; and make a comparison to LNF which is a strictly less succinct than CE-LSNNF. In addition, every language polynomially equivalent to CE-LSNNF satisfies polytime forgetting property.
- For CT-LSNNF (resp. ME-LSNNF), we prove that (1) if a language is polynomially translatable into CT-LSNNF (resp. ME-LSNNF), then it supports polytime model counting (resp. model enumeration), and (2) CT-LSNNF (resp. ME-LSNNF) is polynomially equivalent to d-DNNF (resp. DNNF).

Apart from the above five reasoning tasks, we also apply the notion of logical separability to their duals: validity checking, term implication checking, counter-model counting, counter-model enumeration and ensuring. In a similar way, their corresponding logical separability-based languages: VA-LSNNF, IM-LSNNF, CCT-LSNNF and CME-LSNNF, are identified. Furthermore, we analyze the computational complexity of the membership problems of the proposed languages. We finally analyze the succinctness and tractability of the whole logical separability-based languages, and integrate them into the knowledge compilation map in [9].

A preliminary version of this paper was published in AAAI-2022 [15]. In addition to providing the complete proofs of all propositions, lemmas, and theorems in [15], this article further investigates the application of logical separability to query model enumeration and transformation forgetting as well as their duals.

The rest of the paper is organized as follows. In Section 2, we provide some required preliminaries about the Negation Normal Form (NNF) languages, the notion of succinctness and polynomial translations as well as the queries and transformations properties. In Section 3, we first present a polytime algorithm for satisfiability checking, then propose the definition of CO-logical separability, and investigate the membership complexity of CO-LSNNF. We further study the dual of satisfiability checking: validity checking, and carry out the same research. In Sections 4, 5, 6 and 7, we continue the line of Section 3 for the remaining tasks: clausal entailment checking and term implication checking, model counting and counter-model counting, model enumeration and counter-model enumeration, as well as forgetting and ensuring, respectively. In Section 8, we compare the logical separability-based languages with other languages from the aspects of succinctness, queries and transformations, and propose an extension of the knowledge compilation map. Finally, we offer related works in Section 9, and close with conclusions and future works in Section 10.

2. Preliminaries

2.1. The NNF languages

Throughout this paper, we fix a finite set $\mathbf{X}$ of variables. A *literal* is a variable (positive literal) or a negated one (negative literal). For a positive (resp. negative) literal x (resp. $\neg x$), its complementary literal is $\neg x$ (resp. x). Let $\mathbf{Y} \subseteq \mathbf{X}$. A $\mathbf{Y}$-literal is a literal x or $\neg x$ where $x \in \mathbf{Y}$. The set $\overline{\mathbf{Y}}$ of variables is the complement of $\mathbf{Y}$ w.r.t. $\mathbf{X}$. A *term* (resp. *clause*) is $\top$, $\bot$, or a conjunction (resp. disjunction) of literals. We say a term (resp. clause) is *non-trivial* if every variable appears at most once.

Throughout this paper, we consider a wide range of complete propositional languages, all of which are the subsets of negation normal form (NNF). An NNF-formula is a rooted, directed acyclic graph (DAG), of which each leaf node is labeled by $\top$, $\bot$, or literals; and each internal node is labeled by $\wedge$ (conjunction) or $\vee$ (disjunction).

We use a lower-case Greek letter (e.g. α, β) to denote a propositional formula. We use $\mathrm{Var}(\alpha)$ (resp. $\mathrm{Lit}(\alpha)$) to denote the set of variables (resp. literals) appearing in α. We use $|\alpha|$ to denote the size of α (*i.e.* the number of its DAG edges). For an NNF-formula α, its *complementary formula* $\bar{\alpha}$ is an NNF-formula obtained by replacing any occurrence of Boolean constants $\top$ (resp. $\bot$) by $\bot$ (resp. $\top$), of literals l by its complement $\neg l$ and of connectives $\wedge$ (resp. $\vee$) by $\vee$ (resp. $\wedge$).

A $\mathbf{Y}$-*interpretation* ω is a set of $\mathbf{Y}$-literals s.t. each variable of $\mathbf{Y}$ appears exactly once. Let $\mathrm{Var}(\alpha) \subseteq \mathbf{Y}$. We use $\omega \models \alpha$ (resp. $\omega \not\models \alpha$) to denote ω satisfies (resp. falsifies) α, which is defined as usual. A $\mathbf{Y}$-*model* (resp. $\mathbf{Y}$-*counter-model*) of α is a $\mathbf{Y}$-interpretation satisfying (resp. falsifying) α. We use $\Omega_{\mathbf{Y}}$ for the set of $\mathbf{Y}$-interpretations and $\mathrm{Mod}_{\mathbf{Y}}(\alpha)$ for the set of $\mathbf{Y}$-models of α. For simplicity, in the case $\mathbf{Y} = \mathbf{X}$, we omit the subscript $\mathbf{X}$, and $\mathrm{Mod}(\alpha)$ denotes the set of $\mathbf{X}$-models of α. We say α is *satisfiable* (resp. *falsifiable*), if $\mathrm{Mod}(\alpha) \neq \emptyset$ (resp. $\mathrm{Mod}(\bar{\alpha}) \neq \emptyset$); otherwise, it is *unsatisfiable* (resp. *valid*). A *pseudo-interpretation* is a set of literals that contains a variable x and its complementary literal $\bar{x}$ simultaneously. The Cartesian product on n sets of models $\Omega_1, \cdots, \Omega_n$ is defined as: $\Omega_1 \times \cdots \times \Omega_n = \{\omega_1 \cup \cdots \cup \omega_n \mid \omega_i \in \Omega_i \text{ for } 1 \leq i \leq n\}$.

As mentioned in [9], a language qualifies as a target language if it supports polytime clausal entailment checking. NNF is not a desired target language as it does not support such a query unless $\mathsf{P} = \mathsf{NP}$. But many of its subsets, with one or more restrictions, do.

*Conjunctive normal form (*CNF*)* is the conjunction of non-trivial clauses while its dual, *disjunctive normal form (*DNF*)*, is the disjunction of non-trivial terms. A *prime implicate* c of α is an implicate of α and there is no implicate $c' \neq c$ *s.t.* $\alpha \models c'$ and $c' \models c$. *Prime implicates (*PI*)* is the subset of CNF where each formula α is a conjunction of all of the prime implicates of α. Its dual, *prime implicants (*IP*)*, can be similarly defined.

Decomposable NNF *(*DNNF*)* [13] is the subset of NNF satisfying $\wedge$-decomposability, requiring the sets of variables of the children of each $\wedge$-node in a formula to be pairwise disjoint. *Deterministic* DNNF *(*d-DNNF*)* [16] is the subset of DNNF satisfying determinism, requiring the children of each $\vee$-node in a formula to be pairwise logically contradictory.

KROM [17] is the subset of CNF in which each clause contains at most two literals. HORN [18] is the subset of CNF in which each clause contains at most a positive literal. K/H is the union of KROM and HORN. *Renamable-Horn* (renH) is the subset of all CNF-formulas α for which there is a subset $\mathbf{Y}$ of $\mathrm{Var}(\alpha)$ s.t. the formula obtained by substituting in α every $\mathbf{Y}$-literal l by its complement $\bar{l}$ is a HORN-formula. Fargier and Marquis [19] applied disjunctive closure principle to KROM, HORN, K/H and renH and obtained KROM[$\vee$], HORN[$\vee$], K/H[$\vee$] and renH[$\vee$], respectively.

We remark that the above normal forms are not only defined in purely syntactic style but also based on semantics. For example, CNF and DNF are syntactic normal forms. On the other hand, PI and d-DNNF are based on clausal entailment and satisfiability, respectively and hence being semantic normal forms.

We say that the two languages $\mathcal{L}_1$ and $\mathcal{L}_2$ are dual, if for every $\mathcal{L}_1$-formula α, its complementary formula $\bar{\alpha} \in \mathcal{L}_2$, and vice versa. For instance, CNF and DNF are a pair of dual languages, and so are PI and IP.

2.2. Succinctness and polynomial translations

We now consider two notions of translations on two subsets of NNF: *succinctness* and *polynomial-translation*.

Definition 1. Let $\mathcal{L}_1$ and $\mathcal{L}_2$ be subsets of NNF. We say

- $\mathcal{L}_1$ is at least as succinct as $\mathcal{L}_2$, denoted $\mathcal{L}_1 \leq_s \mathcal{L}_2$, if there is a polynomial p s.t. for every formula $\alpha \in \mathcal{L}_2$, there is an equivalent formula $\beta \in \mathcal{L}_1$ s.t. $|\beta| \leq p(|\alpha|)$.
- $\mathcal{L}_2$ is polynomially translatable into $\mathcal{L}_1$, denoted $\mathcal{L}_1 \leq_p \mathcal{L}_2$, if there exists a (deterministic) polynomial-time algorithm f s.t. for every formula $\alpha \in \mathcal{L}_2$, we have an equivalent formula $f(\alpha) \in \mathcal{L}_1$.

Succinctness only considers polynomial-space translations, that is, the size of the $\mathcal{L}_1$-formula β equivalent to α is required to be polynomial in the size of $\mathcal{L}_2$-formula α. Polynomial-translation is a more strict relation than succinctness, requiring polynomial-time translations, that is, any $\mathcal{L}_2$-formula can be tractably transformed into an equivalent $\mathcal{L}_1$-formula. Furthermore, we have $\mathcal{L}_1 \leq_p \mathcal{L}_2$ implies that $\mathcal{L}_1 \leq_s \mathcal{L}_2$ [19].

The two relations $\leq_s$ and $\leq_p$ are clearly reflexive and transitive, *i.e.* pre-orders over subsets of NNF. The notation $\sim_s$ denotes the symmetric part of $\leq_s$, that is, $\mathcal{L}_1 \sim_s \mathcal{L}_2$ iff $\mathcal{L}_1 \leq_s \mathcal{L}_2$ and $\mathcal{L}_2 \leq_s \mathcal{L}_1$. On the other hand, $<_s$ denotes the asymmetric part of $\leq_s$, that is, $\mathcal{L}_1 <_s \mathcal{L}_2$ iff $\mathcal{L}_1 \leq_s \mathcal{L}_2$ and $\mathcal{L}_2 \not\leq_s \mathcal{L}_1$. In the following, $\mathcal{L}_1 \not\leq_s^* \mathcal{L}_2$ means that $\mathcal{L}_1 \not\leq_s \mathcal{L}_2$ unless the polynomial hierarchy PH collapses. The notations $\sim_p$, $<_p$ and $\not\leq_p^*$ can be similarly defined.

2.3. Queries and transformations

As pointed out in [9], another two crucial criteria for evaluating normal forms is the set of query and transformation tasks supported in polytime. To differentiate query (or transformation) tasks and properties, we use letters in typewriter type font for the former, and letters in the boldface font for the latter.

A *query task* for a language $\mathcal{L}$ takes as input one or more $\mathcal{L}$-formulas and possibly clauses and terms, and outputs a Boolean value, a natural number, or a set of interpretations. We consider the following query tasks:

- Satisfiability (CO): check if $\alpha \not\equiv \bot$.
- Validity (VA): check if $\alpha \equiv \top$.
- Clausal entailment (CE): check if $\alpha \models c$ where c is a clause.
- Term implication (IM): check if $t \models \alpha$ where t is a term.
- Sentential entailment (SE): check if $\alpha \models \beta$.
- Equivalence (EQ): check if $\alpha \equiv \beta$.
- Model counting (CT): compute the size of $\text{Mod}(\alpha)$.
- Counter-model counting (CCT): compute the size of $\text{Mod}(\bar{\alpha})$.
- Model enumeration (ME): list every member of $\text{Mod}(\alpha)$.
- Counter-model enumeration (CME): list every member of $\text{Mod}(\bar{\alpha})$.

A *transformation task* for $\mathcal{L}$ takes as input one or a set of $\mathcal{L}$-formulas, possibly a term and a set of variables, and returns an appropriate $\mathcal{L}$-formula. We consider the following transformation tasks:

- Conditioning (CD): generate $\alpha|t$ where t is a satisfiable term;
- Forgetting (FO): generate $\exists \mathbf{Y}.\alpha$ where $\mathbf{Y} \subseteq \mathbf{X}$.
- Singleton forgetting (SFO): generate $\exists \{x\}.\alpha$ where $x \in \mathbf{X}$.
- Ensuring (EN): generate $\forall \mathbf{Y}.\alpha$ where $\mathbf{Y} \subseteq \mathbf{X}$.
- Singleton ensuring (SEN): generate $\forall \{x\}.\alpha$ where $x \in \mathbf{X}$.
- Conjunction ($\wedge$C): generate $\alpha_1 \wedge \cdots \wedge \alpha_n$.
- Bounded conjunction ($\wedge$BC): generate $\alpha \wedge \beta$.
- Disjunction ($\vee$C): generate $\alpha_1 \vee \cdots \vee \alpha_n$.
- Bounded disjunction ($\vee$BC): generate $\alpha \vee \beta$.
- Negation ($\neg$C): generate $\neg\alpha$.

Specifically, conditioning a formula α on a satisfiable term t, denoted by $\alpha|t$, is the result of replacing each occurrence of a variable x in α by $\top$ (resp. $\bot$), if x (resp. $\neg x$) is a positive (resp. negative) literal of t. Forgetting $\mathbf{Y}$ from α, denoted as $\exists \mathbf{Y}.\alpha$, is the strongest consequence that does not contain any variables of $\mathbf{Y}$, that is, for any NNF-formula β s.t. $\text{Var}(\beta) \cap \mathbf{Y} = \emptyset$, $\alpha \models \beta$ iff $\exists \mathbf{Y}.\alpha \models \beta$. Similarly, ensuring $\mathbf{Y}$ from α, denoted as $\forall \mathbf{Y}.\alpha$, is the weakest antecedent that does not contain any variables of $\mathbf{Y}$, that is, for any NNF-formula β s.t. $\text{Var}(\beta) \cap \mathbf{Y} = \emptyset$, $\beta \models \alpha$ iff $\beta \models \forall \mathbf{Y}.\alpha$. We say a language $\mathcal{L}$ satisfies the query property $\mathbf{Q}$, iff there is a polytime procedure for the query task Q for $\mathcal{L}$. We say a language $\mathcal{L}$ satisfies the transformation property $\mathbf{T}$, iff there is a polytime procedure for the transformation task T for $\mathcal{L}$ and the output formula of the procedure is in $\mathcal{L}$. For example, DNF supports **CO** since the CO task can be performed under DNF in polytime [9].

3. Satisfiability and validity

3.1. Satisfiability

We first present a procedure for satisfiability checking on NNF-formulas, which was proposed by Darwiche [13].

Definition 2. The procedure $\mathcal{CO}(\alpha)$ is recursively defined as:

- $\mathcal{CO}(\top) = 1$, $\mathcal{CO}(\bot) = 0$ and $\mathcal{CO}(l) = 1$
- $\mathcal{CO}(\beta_1 \wedge \cdots \wedge \beta_n) = \min_{1 \le i \le n} \{\mathcal{CO}(\beta_i)\}$
- $\mathcal{CO}(\beta_1 \vee \cdots \vee \beta_n) = \max_{1 \le i \le n} \{\mathcal{CO}(\beta_i)\}$

The above procedure works in a recursive way. In the base case, the Boolean constant $\top$ and a literal l are satisfiable while $\bot$ is unsatisfiable. In the induction case, the satisfiability problem of an $\vee$-node (or $\wedge$-node) is reduced to the same problem of its subformula β_i. If some β_i of $\vee$-node α is satisfiable, then α is satisfiable. Similarly, if every β_i of $\wedge$-node α is satisfiable, then α is satisfiable.

Definition 3. Let $\mathcal{L}$ be a language, and f a procedure that takes an NNF-formula α as input, and that returns a Boolean value. We say

- f is CO-sound for $\mathcal{L}$ iff for every $\mathcal{L}$-formula α, $f(\alpha) = 1$ only if $\mathtt{CO}(\alpha) = 1$;
- f is CO-complete for $\mathcal{L}$ iff for every $\mathcal{L}$-formula α, $\mathtt{CO}(\alpha) = 1$ only if $f(\alpha) = 1$.

The procedure $\mathcal{CO}$ is a CO-complete algorithm for any NNF-formula α and it can be evaluated at Linear time in $|\alpha|$.

Theorem 1. *[13] The procedure $\mathcal{CO}$ is CO-complete for NNF with the time complexity $O(|\alpha|)$.*

However, as shown in Example 1, the procedure $\mathcal{CO}$ is not CO-sound for the NNF language.

Example 1. Consider an NNF-formula $\alpha = x \wedge \neg x$. Clearly, α is unsatisfiable, *i.e.*, $\mathtt{CO}(\alpha) = 0$. By the definition of procedure $\mathcal{CO}$, $\mathcal{CO}(x) = \mathcal{CO}(\neg x) = 1$. It follows that $\mathcal{CO}(\alpha) = 1$ which contradicts the unsatisfiability of α. □

The reason why the procedure $\mathcal{CO}$ cannot serve as a sound algorithm is that $\mathcal{CO}$ is a simple evaluation-based algorithm and it does not consider implicit logical contradictions occurring in some $\wedge$-nodes of an NNF-formula. Formally, we say an $\wedge$-node $\alpha = \beta_1 \wedge \cdots \wedge \beta_n$ contains an implicit logical contradiction, if α is unsatisfiable and each β_i is satisfiable for $1 \leq i \leq n$. In this case, $\mathcal{CO}(\alpha) = 1$ as $\mathcal{CO}(\beta_i) = 1$ for $1 \leq i \leq n$, and hence deriving an incorrect result.

We hereafter provide a necessary and sufficient condition on NNF for which the $\mathcal{CO}$ procedure is complete and sound.

Definition 4. An NNF-formula α is CO-logically separable ($\mathtt{CO}_{ls}$) iff

- α is $\top$, $\bot$, or a literal l; or
- α is an $\wedge$-node $\beta_1 \wedge \cdots \wedge \beta_n$, and if α is unsatisfiable, then β_i is unsatisfiable and $\mathtt{CO}_{ls}$ for some $1 \leq i \leq n$; or
- α is an $\vee$-node $\beta_1 \vee \cdots \vee \beta_n$, and if α is unsatisfiable, then β_i is unsatisfiable and $\mathtt{CO}_{ls}$ for every $1 \leq i \leq n$.

We use CO-LSNNF to denote the subset of NNF in which any formula satisfies CO-logical separability property. To avoid the implicit logical contradictions in an unsatisfiable $\wedge$-node $\beta_1 \wedge \cdots \wedge \beta_n$, Definition 4 requires some conjuncts β_i to be unsatisfiable and CO-logically separable. Under this requirement, logical contradictions of $\wedge$-nodes occurs explicitly so that we can easily refute the satisfiability of α. In a similar way, each disjunct of an unsatisfiable $\vee$-node is unsatisfiable and CO-logically separable. Hence, we obtain the soundness of the procedure $\mathcal{CO}$. Example 2 illustrates the CO-logical separability property.

Example 2 *(Example 1 cont'd).* Recall the unsatisfiable formula $\alpha = x \wedge \neg x$. None of its two conjuncts x and $\neg x$ is unsatisfiable. Therefore, α does not satisfy CO-logical separability property. Consider now its variant $\alpha' = x \wedge \neg x \wedge \bot$, which is also unsatisfiable. It contains a conjunct $\bot$ that is both unsatisfiable and CO-logically separable. Hence α' is in CO-LSNNF, and contains no implicit logical contradictions.

Note that CO-logical separability property is not to eliminate all implicit logical contradictions in all $\wedge$-nodes of a given formula. Consider a CO-LSNNF-formula $\alpha_1 = ((x \wedge \neg x) \vee y) \wedge \bot$. The logical contradiction hidden in the inner $\wedge$-node $x \wedge \neg x$ does not impede the correctness of the procedure $\mathcal{CO}$ due to the occurrence of $\bot$ in the outermost $\wedge$-node of α_1. □

The following theorem shows that CO-logical separability property is a sufficient and necessary condition to guarantee correctness of the procedure $\mathcal{CO}$.

Theorem 2. *A language $\mathcal{L}$ is in CO-LSNNF iff the procedure $\mathcal{CO}$ is CO-complete and CO-sound for $\mathcal{L}$.*

We observe that all normal forms, summarized in [9], supporting **CO** include PI, DNNF, OBDD and so on. Interestingly, they are either PI or a subset of DNNF. In addition, we observe that every formula of PI or DNNF satisfies CO-logical separability.

Proposition 1. *The languages DNNF and PI are subsets of CO-LSNNF.*

On the one side, $\wedge$-decomposability forbids simultaneous occurrence of the same variable in distinct conjuncts of any $\wedge$-node. Hence, no implicit logical contradiction occurs in $\wedge$-nodes of a DNNF-formula. On the other side, PI requires that any implicate of α is derived by a clause c of α and that no implicate $c' \neq c$ of α such that $c' \models c$. The Boolean constant $\bot$ is a disjunction of an empty set of literals. If α implies $\bot$, then it must be $\bot$ itself. The logical contradiction therefore appears explicitly in any unsatisfiable PI-formula.

Someone may wonder if the two properties polytime satisfiability checking and CO-logical separability are essentially equivalent? The four normal forms: KROM[$\vee$], HORN[$\vee$], renH[$\vee$] and K/H[$\vee$], which supports polytime satisfiability checking [19], are counterexamples for the above question. As mentioned before, the formula $x \wedge \neg x$ is not CO-logically separable. Clearly, it is a formula of the above four normal forms.

The story does not come to the end. We connect these two properties via polytime translation.

Theorem 3. *A language $\mathcal{L}$ satisfies* **CO** *iff it is polynomially translatable into* CO-LSNNF.

In other words, $\mathcal{L}$ supports polytime satisfiability checking iff the implicit logically contradictions in any $\mathcal{L}$-formula α that are witnesses to refute the satisfiability of α can be efficiently discovered.

We close this section by comparing CO-LSNNF and NNF in terms of succinctness and polynomial-translation.

Proposition 2.

- NNF $\sim_s$ CO-LSNNF
- NNF $\leq_p$ CO-LSNNF *and* CO-LSNNF $\not\leq_p^*$ NNF.

As mentioned in [19], the inclusion $\leq_p \subset \leq_s$ holds. It means that there are two languages $\mathcal{L}_1$ and $\mathcal{L}_2$ such that $\mathcal{L}_1 \leq_s \mathcal{L}_2$ holds but $\mathcal{L}_1 \leq_p \mathcal{L}_2$ does not. However, they do not provide two languages confirming the inclusion $\leq_p \subset \leq_s$. We have surveyed the existing literature regarding succinctness among various propositional representations [9,20–22,11,19,23,24]. We find that in these literature, for every two languages $\mathcal{L}_1$ and $\mathcal{L}_2$, if the succinctness relation $\mathcal{L}_1 \leq_s \mathcal{L}_2$ holds, then the polynomial-translation relation $\mathcal{L}_1 \leq_p \mathcal{L}_2$ also holds. To the best of our knowledge, this paper is the first one to provide the pair of languages which is a witness to the inclusion $\leq_p \subset \leq_s$.

A classical knowledge compilation result in [25] states that unless PH collapses, there does not exist a class $\mathcal{L}$ of formulas s.t. every NNF-formula has a polysize equivalent $\mathcal{L}$-formula, and $\mathcal{L}$ supports polytime clausal entailment. It is well-known that both CE and CO tasks for NNF are reasoning problems with high computational complexity, which are coNP-complete and NP-complete, respectively. In this paper, we show that CO-LSNNF is the language that supports polytime satisfiability checking and that is equivalent succinct to NNF, proving the similar result in [25] for satisfiability checking does not hold.

Finally, determining if an NNF-formula is CO-logically separable is the same as hard as the satisfiability problem.

Proposition 3. *Deciding if an* NNF*-formula is in* CO-LSNNF *is* NP*-complete.*

3.2. Validity

We now focus on another well-known query: validity, which is exactly the dual one of satisfiability. We obtain a sound procedure $\mathcal{VA}$ for validity checking from the procedure $\mathcal{CO}$ via a slight modification: $\mathcal{VA}(l) = 0$.

Definition 5. Let $\mathcal{L}$ be a language, and f a procedure that takes an NNF-formula α as input, and that returns a Boolean value. We say

- f is VA-sound for $\mathcal{L}$ iff for every $\mathcal{L}$-formula α, $f(\alpha) = 1$ only if $\mathtt{VA}(\alpha) = 1$;
- f is VA-complete for $\mathcal{L}$ iff for every $\mathcal{L}$-formula α, $\mathtt{VA}(\alpha) = 1$ only if $f(\alpha) = 1$.

Similarly to $\mathcal{CO}$, we offer a necessary and sufficient condition under which the procedure $\mathcal{VA}$ is VA-complete and VA-sound.

Definition 6. An NNF-formula α is VA-logically separable ($\mathtt{VA}_{ls}$) iff

- α is $\top$, $\bot$, or a literal l; or
- α is an $\wedge$-node $\beta_1 \wedge \cdots \wedge \beta_n$, and if α is valid, then β_i is valid and $\mathtt{VA}_{ls}$ for every $1 \leq i \leq n$; or
- α is an $\vee$-node $\beta_1 \vee \cdots \vee \beta_n$, and if α is valid, then β_i is valid and $\mathtt{VA}_{ls}$ for some $1 \leq i \leq n$.

For a valid $\vee$-node, VA-logical separability property requires at least one of its disjunct to be valid and $\mathtt{VA}_{ls}$. As to a valid $\wedge$-node, all of its conjuncts are required to be valid and $\mathtt{VA}_{ls}$.

Denote VA-LSNNF to the subset of NNF in which any formula satisfies VA-logical separability property. Clearly, the language VA-LSNNF is a dual language to CO-LSNNF.

Example 3 provides a VA-LSNNF-formula and illustrates the run of procedure $\mathcal{VA}$ on the formula.

Example 3. Consider the formula $\alpha = (x \vee \neg x \vee \top) \wedge (y \vee \top)$. By Definition 6, literals x, $\neg x$ and y and Boolean constant $\top$ are VA-logically separable. Moreover, the disjunctions $x \vee \neg x \vee \top$ and $y \vee \top$ are also VA-logically separable, since they both have a valid and VA-logically separable disjunct $\top$. Therefore, α satisfies VA-logical separability.

The procedure $\mathcal{VA}$ on α works as follows. To begin with, $\mathcal{VA}(x) = \mathcal{VA}(\neg x) = \mathcal{VA}(y) = 0$ and $\mathcal{VA}(\top) = 1$. Then, since $\mathcal{VA}(\top) = 1$, we have that $\mathcal{VA}(x \vee \neg x \vee \top) = \mathcal{VA}(y \vee \top) = 1$. Finally, we get that the output $\mathcal{VA}(\alpha) = 1$, is consistent with the fact that $\mathtt{VA}(\alpha) = 1$. □

We show that (1) the procedure $\mathcal{VA}$ is a polytime algorithm that is VA-sound for arbitrary NNF-formula; (2) the language VA-LSNNF precisely captures the maximal fragment of NNF to guarantee the correctness of the procedure $\mathcal{VA}$; (3) a language $\mathcal{L}$ permits **VA** if and only if it is polynomially translatable into VA-LSNNF.

Theorem 4. *The procedure $\mathcal{VA}$ is* VA*-sound for* NNF *with the time complexity* $O(|\alpha|)$.

Theorem 5. *A language $\mathcal{L}$ is in* VA-LSNNF *iff the procedure $\mathcal{VA}$ is* VA*-complete and* VA*-sound for $\mathcal{L}$.*

Theorem 6. *A language $\mathcal{L}$ satisfies* **VA** *iff it is polynomially translatable into* VA-LSNNF.

Finally, we claim that (1) the language VA-LSNNF is equally succinct to NNF; (2) determining if an NNF-formula is VA-logically separable is NP-complete.

Proposition 4. VA-LSNNF $\sim_s$ NNF.

Proposition 5. *Deciding if an* NNF*-formula is in* VA-LSNNF *is* NP*-complete.*

4. Clausal entailment and term implication

4.1. Clausal entailment

As mentioned in [9], a language qualifies as a target language, if it permits polytime clausal entailment checking. In this section, we consider logical separability in the query clausal entailment. We begin with providing a recursive procedure for clausal entailment as follows.

Definition 7. Let α be an NNF-formula and c a non-trivial clause. The procedure $\mathcal{CE}(\alpha, c)$ is recursively defined as:

- $\mathcal{CE}(\top, c) = \begin{cases} 1, & \text{if } c = \top \\ 0, & \text{otherwise} \end{cases}$
- $\mathcal{CE}(\bot, c) = 1$, $\mathcal{CE}(l, \top) = 1$ and $\mathcal{CE}(l, \bot) = 0$
- $\mathcal{CE}(l, l_1 \vee \cdots \vee l_m) = \begin{cases} 1, & \text{if } l = l_i \text{ for some } i \\ 0, & \text{otherwise} \end{cases}$
- $\mathcal{CE}(\beta_1 \wedge \cdots \wedge \beta_n, c) = \max\limits_{1 \le i \le n} \{\mathcal{CE}(\beta_i, c)\}$
- $\mathcal{CE}(\beta_1 \vee \cdots \vee \beta_n, c) = \min\limits_{1 \le i \le n} \{\mathcal{CE}(\beta_i, c)\}$

The above procedure is a recursive algorithm. In the case that α is a Boolean constant $\top$, the procedure returns 1 if the clause c is also $\top$, and returns 0 otherwise. In the case that α is $\bot$, the procedure always returns 1 since an unsatisfiable KB derives any consequence. In the case that α is a literal l, then the procedure returns 1 if l is a literal of the clause c, or c is $\top$ since l entails a clause including l and $\top$ is a consequence of any formula. The clausal entailment problem of an $\wedge$-node $\beta_1 \wedge \cdots \wedge \beta_n$ and a clause c can be reduced to the same subproblem of β_i and c. If one of the β_i's entails c, then their conjunction does. The case that α is an $\vee$-node is similarly handled.

Darwiche [13] proposed a polytime clausal entailment procedure $\mathcal{CE}'$ via the procedure $\mathcal{CO}$ and conditioning. The definition of $\mathcal{CE}'$ is as follows: $\mathcal{CE}'(\alpha, c) = 1$ iff $\mathcal{CO}(\alpha|t) = 0$ where t is equivalent to $\neg c$. In fact, the two procedures $\mathcal{CE}$ and $\mathcal{CE}'$ are equivalent, that is, they return the same result for every formula α and every non-trivial clause c.

Proposition 6. *Let α be an* NNF*-formula and c a non-trivial clause. Then, $\mathcal{CE}(\alpha, c) = \mathcal{CE}'(\alpha, c)$.*

The major advantage of the procedure $\mathcal{CE}$ over $\mathcal{CE}'$ is that the former directly solves the clausal entailment problem while the latter resorts to two procedures for satisfiability checking and conditioning.

Definition 8. Let $\mathcal{L}$ be a language and f be a procedure that takes an NNF-formula α and a non-trivial clause c as input, and that returns a Boolean value. We say

- f is CE-sound for $\mathcal{L}$ iff for every $\mathcal{L}$-formula α and every non-trivial clause c, $f(\alpha, c) = 1$ only if $\mathrm{CE}(\alpha, c) = 1$;
- f is CE-complete for $\mathcal{L}$ iff for every $\mathcal{L}$-formula α and every non-trivial clause c, $\mathrm{CE}(\alpha, c) = 1$ only if $f(\alpha, c) = 1$.

The procedure $\mathcal{CE}$ is a sound algorithm for clausal entailment and takes polytime in the size of α and c.

Theorem 7. *The procedure $\mathcal{CE}$ is* CE*-sound for* NNF *with the time complexity* $O(|\alpha| \cdot |c|)$.

However, it is not guaranteed that the procedure $\mathcal{CE}$ is complete if α is an arbitrary NNF-formula. We illustrate this with the following example.

Example 4. Consider the formula $\alpha = (\neg x \vee y) \wedge (\neg y \vee z)$ and the clause $c = \neg x \vee z$. It is easily verified that $\mathcal{CE}(\neg x, c) = 1$, $\mathcal{CE}(y, c) = 0$, $\mathcal{CE}(\neg y, c) = 0$ and $\mathcal{CE}(z, c) = 1$. It follows that $\mathcal{CE}(\neg x \vee y, c) = 0$ and $\mathcal{CE}(\neg y \vee z, c) = 0$. Finally, $\mathcal{CE}(\alpha, c) = 0$. On the contrary, $\mathtt{CE}(\alpha, c) = 1$ as $\alpha \models c$. □

The reason that the procedure $\mathcal{CE}$ cannot give a complete answer for clausal entailment is similar to why the $\mathcal{CO}$ does not work perfectly for satisfiability checking. The procedure $\mathcal{CE}$ only decomposes $\wedge$-nodes of a formula α in a simple way and does not reasoning about implicit logical implicate of these $\wedge$-nodes. We say an $\wedge$-node $\alpha = \beta_1 \wedge \cdots \wedge \beta_n$ contains an implicit logical implicate, if there is a clause c s.t. $\alpha \models c$ and $\beta_i \not\models c$ for every $1 \leq i \leq n$.

Based on the above observations, we hereafter give a normal form that is the sufficient and necessary condition of making the procedure $\mathcal{CE}$ not only sound but also complete.

Definition 9. Let c be a non-trivial clause. An NNF-formula α is CE-logically separable ($\mathtt{CE}_{ls}$) w.r.t. c iff

- α is $\top$, $\bot$, or a literal l; or
- α is an $\wedge$-node $\beta_1 \wedge \cdots \wedge \beta_n$, and if $\alpha \models c$, then there is a conjunct β_i s.t. $\beta_i \models c$ and β_i is $\mathtt{CE}_{ls}$ w.r.t. c; or
- α is an $\vee$-node $\beta_1 \vee \cdots \vee \beta_n$, and if $\alpha \models c$, then for every disjunct β_i, we have $\beta_i \models c$ and β_i is $\mathtt{CE}_{ls}$ w.r.t. c.

The Boolean constants and a literal l are $\mathtt{CE}_{ls}$ w.r.t. any non-trivial clause c. Suppose that α is an $\wedge$-node $\beta_1 \wedge \cdots \wedge \beta_n$ which entails a clause c. The clause c may be an implicit logical implicate of α. To avoid them, Definition 9 requires some of its conjunct to entail c and to be CE-logically separable w.r.t. c. In the similar way, if an $\vee$-node entails clause c, then each of its disjuncts also entails c and is CE-logically separable w.r.t. c.

We say an NNF-formula satisfies general CE-logically separable (general $\mathtt{CE}_{ls}$), if it is $\mathtt{CE}_{ls}$ w.r.t. every non-trivial clause. We use CE-LSNNF to denote the subset of NNF for which every formula satisfies general $\mathtt{CE}_{ls}$. General CE-logical separability eliminates any implicit implicate of some $\wedge$-nodes so as to gain the completeness of the procedure $\mathcal{CE}$ for every formula and every non-trivial clause.

Example 5 provides a formula that is equivalent to the formula α in Example 4 and that satisfies general CE-logical separability.

Example 5 *(Example 4 cont'd).* Recall the NNF-formula $\alpha = (\neg x \vee y) \wedge (\neg y \vee z)$ and the clause $c = \neg x \vee z$. By conjoining α with the clause $\neg x \vee z$, the new formula $\alpha' = (\neg x \vee y) \wedge (\neg y \vee z) \wedge (\neg x \vee z)$ contains no implicit logical implicates. We now examine the procedure $\mathcal{CE}$ for the formula α', which is equivalent to the previous one α. It is easily verified that $\mathcal{CE}(\neg x \vee z, c) = 1$, and hence $\mathcal{CE}(\alpha', c) = 1$. The procedure $\mathcal{CE}(\alpha', c)$ generates a result in accordance with $\mathtt{CE}(\alpha', c)$.

Actually, the formula α' is in CE-LSNNF. First of all, it holds that α' entails c. Besides, the conjunct $\neg x \vee z$ of α' also entails c, and is $\mathtt{CE}_{ls}$ w.r.t. c. Therefore, we can get that α' is CE-logically separable w.r.t. the clause c. Similarly, it can be verified that α' is CE-logically separable w.r.t. any other non-trivial clause. □

Theorem 8. *A language $\mathcal{L}$ is* CE-LSNNF *iff the procedure $\mathcal{CE}$ is* CE*-complete and* CE*-sound for $\mathcal{L}$.*

We observe that $\wedge$-decomposability and prime implicates are special cases of general CE-logical separability.

Proposition 7. DNNF *and* PI *are subsets of* CE-LSNNF.

We elaborate the intuition behind $\wedge$-decomposability and prime implicant via the inference rule: resolution. Resolution is used to find a refutation proof of CNF-formulas [26]. Levesque [14] used resolution to discover implicit implicates hidden in $\wedge$-nodes. The conjunction of $x \vee c_1$ and $\neg x \vee c_2$ deduces their resolvent $c_1 \vee c_2$. This deduction also holds if the two clauses c_1 and c_2 are replaced by two arbitrary formulas β_1 and β_2 respectively.

On the one side, $\wedge$-decomposability precludes the case that a variable x simultaneously occurs in different conjuncts of an $\wedge$-node α. This causes no occurrence of implicit implicate within any $\wedge$-node. On the other side, PI-formula α requires every resolution among every two clauses of α to be entailed by a clause within α. Hence, no implicit implicate occurs via making advantage of resolution.

It is easily verified that every language $\mathcal{L}$ that is polynomially translatable into CE-LSNNF also satisfies **CE**.

Theorem 9. *If a language $\mathcal{L}$ is polynomially translatable into* CE-LSNNF*, then $\mathcal{L}$ satisfies* **CE**.

But the opposite direction remains open. Considering the four disjunctive closure based languages, none of these languages is at least as succinct as CE-LSNNF. Any KROM-formula can be converted into a PI-formula in polytime [27]. In addition, PI is a subset of CE-LSNNF and CE-LSNNF supports polytime disjunction ($\vee$**C**), which will be shown in Table 1. We therefore obtain that KROM[$\vee$] is polynomially translatable into CE-LSNNF. So CE-LSNNF is strictly more succinct than KROM[$\vee$]. The problem whether CE-LSNNF is at least as succinct as renH[$\vee$], K/H[$\vee$] and HORN[$\vee$] remains unknown.

Every CO-LSNNF-formula is an NNF-formula that is CE-logically separable w.r.t. $\bot$. So CO-LSNNF $\leq_s$ CE-LSNNF. But the opposite direction does not hold unless PH collapses.

Proposition 8. CO-LSNNF $\leq_s$ CE-LSNNF *and* CE-LSNNF $\not\leq_s^*$ CO-LSNNF.

CO-LSNNF does not support polytime conditioning which will be shown in Table 1. The following proposition states that CE-LSNNF is the maximal fragment of CO-LSNNF closed under conditioning.

Proposition 9. *An* NNF*-formula* $\alpha \in$ CE-LSNNF *iff* $\alpha \in$ CO-LSNNF *and* $\alpha|t \in$ CO-LSNNF *for every non-trivial term* t.

The following proposition provides the upper bound of the CE-LSNNF membership problem.

Proposition 10. *Deciding if an* NNF*-formula is in* CE-LSNNF *is in* Π_2^p.

4.2. Term implication

Now, let us concentrate on the dual task of clausal entailment: term implication. Given an arbitrary NNF-formula α and a term t, we have that $t \vDash \alpha$ if and only if $\alpha' \vDash c$, where α' and c are complementary formulas of α and t respectively. Utilizing this equivalent condition, a recursive procedure $\mathcal{IM}$ for term implication can be designed with a simple modification of $\mathcal{CE}$.

Definition 10. Let α be an NNF-formula and t a non-trivial term. The procedure $\mathcal{IM}(\alpha, t)$ is recursively defined as:

- $\mathcal{IM}(\bot, t) = \begin{cases} 1, & \text{if } t = \bot \\ 0, & \text{otherwise} \end{cases}$
- $\mathcal{IM}(\top, t) = 1$, $\mathcal{IM}(l, \bot) = 1$ and $\mathcal{IM}(l, \top) = 0$
- $\mathcal{IM}(l, l_1 \wedge \cdots \wedge l_m) = \begin{cases} 1, & \text{if } l = l_i \text{ for some } i \\ 0, & \text{otherwise} \end{cases}$
- $\mathcal{IM}(\beta_1 \wedge \cdots \wedge \beta_n, t) = \min\limits_{1 \leq i \leq n} \{\mathcal{IM}(\beta_i, t)\}$
- $\mathcal{IM}(\beta_1 \vee \cdots \vee \beta_n, t) = \max\limits_{1 \leq i \leq n} \{\mathcal{IM}(\beta_i, t)\}$

Definition 11. Let $\mathcal{L}$ be a language and f be a procedure that takes an NNF-formula α and a non-trivial term t as input, and that returns a Boolean value. We say

- f is IM-sound for $\mathcal{L}$ iff for every $\mathcal{L}$-formula α and every non-trivial term t, $f(\alpha, t) = 1$ only if $\mathtt{IM}(\alpha, t) = 1$;
- f is IM-complete for $\mathcal{L}$ iff for every $\mathcal{L}$-formula α and every non-trivial term t, $\mathtt{IM}(\alpha, t) = 1$ only if $f(\alpha, t) = 1$.

The procedure $\mathcal{IM}$ is IM-sound for NNF and takes polytime in the size of α and t.

Theorem 10. *The procedure* $\mathcal{IM}$ *is* IM*-sound for* NNF *with the time complexity* $O(|\alpha| \cdot |t|)$.

We hereafter provide a sufficient and necessary condition to guarantee the correctness of the procedure $\mathcal{IM}$.

Definition 12. Let t be a non-trivial term. An NNF-formula α is IM-logically separable ($\mathtt{IM}_{ls}$) w.r.t. t iff

- α is $\top$, $\bot$, or a literal l; or
- α is an $\wedge$-node $\beta_1 \wedge \cdots \wedge \beta_n$, and if $t \vDash \alpha$, then for every conjunct β_i, we have $t \vDash \beta_i$ and β_i is $\mathtt{IM}_{ls}$ w.r.t. t; or
- α is an $\vee$-node $\beta_1 \vee \cdots \vee \beta_n$, and if $t \vDash \alpha$, then there is a disjunct β_i s.t. $t \vDash \beta_i$ and β_i is $\mathtt{IM}_{ls}$ w.r.t. t.

For an $\vee$-node entailed by t, the above constraint requires some of its disjunct β_i to be entailed by t and to be $\mathtt{IM}_{ls}$ w.r.t. t. The constraint for an $\wedge$-node is similar, except that we take all β_i's into consideration.

An NNF-formula is said to be general IM-logically separable (general $\mathtt{IM}_{ls}$), if it is $\mathtt{IM}_{ls}$ w.r.t. every non-trivial term. We use IM-LSNNF to denote the subset of NNF for which every formula satisfies general $\mathtt{IM}_{ls}$. It is easy to verify that the language IM-LSNNF is a dual language to CE-LSNNF. The following example provides an IM-LSNNF-formula, for which term implication checking can be correctly solved by procedure $\mathcal{IM}$.

Example 6. Given a formula $\alpha = (\neg x \wedge y) \vee (\neg y \wedge z) \vee (\neg x \wedge z)$ and a term $t = \neg x \wedge z$. Clearly, $t \vDash \alpha$. In addition, t entails the disjunct $\neg x \wedge z$ of α, and $\neg x \wedge z$ is $\mathtt{IM}_{ls}$ w.r.t. t. Therefore, α is IM-logically separable w.r.t. the term t. Besides, it can be verified that α is IM-logically separable w.r.t. any other non-trivial term. Hence, α satisfies general IM-logical separability.

Consider now the run of procedure $\mathcal{IM}$ on the formula α and the term t. By Definition 10, $\mathcal{IM}(\neg x, t) = \mathcal{IM}(z, t) = 1$ and $\mathcal{IM}(y, t) = \mathcal{IM}(\neg y, t) = 0$. Moreover, we get that $\mathcal{IM}(\neg x \wedge y, t) = \mathcal{IM}(\neg y \wedge z, t) = 0$ and $\mathcal{IM}(\neg x \wedge z, t) = 1$. These yield $\mathcal{IM}(\alpha, t) = 1$, which coincides with the fact that $\mathtt{IM}(\alpha, t) = 1$. □

We claim that (1) general IM-logical separability property is a sufficient and necessary condition to guarantee the correctness of the procedure $\mathcal{IM}$; (2) every language $\mathcal{L}$ that is polynomially translatable into IM-LSNNF also satisfies **IM**.

Theorem 11. *A language $\mathcal{L}$ is* IM-LSNNF *iff the procedure $\mathcal{IM}$ is* IM*-complete and* IM*-sound for $\mathcal{L}$.*

Theorem 12. *If a language $\mathcal{L}$ is polynomially translatable into* IM-LSNNF*, then $\mathcal{L}$ satisfies* **IM**.

At last, we show that (1) VA-LSNNF is strictly succinct than IM-LSNNF unless PH collapses; (2) IM-LSNNF and CE-LSNNF are incomparable w.r.t. succinctness unless PH collapses; (3) the upper bound of the IM-LSNNF membership problem is Π_2^p.

Proposition 11.

- VA-LSNNF $\leq_s$ IM-LSNNF *and* IM-LSNNF $\not\leq_s^*$ VA-LSNNF.
- IM-LSNNF $\not\leq_s^*$ CE-LSNNF *and* CE-LSNNF $\not\leq_s^*$ IM-LSNNF.

Proposition 12. *Deciding if an* NNF*-formula is in* IM-LSNNF *is in Π_2^p.*

4.3. Comparison to Levesque's normal form

In first-order logic, Levesque [14] defined a class for queries, namely Levesque's normal norm (LNF), such that entailment is complete, sound and can be tractably solved for a class of knowledge base, namely proper knowledge base, which is equivalent to a possibly infinite consistent set of ground literals. Since the negation $\neg$ is applied to a formula, LNF is not a subset of NNF. To make a clear comparison of LNF with CE-LSNNF and IM-LSNNF, we propose the NNF version of LNF, namely Levesque's negation normal norm (LNNF). We remark that every LNF-formula can be converted into a polysize equivalent one in LNNF, and vice versa.

Definition 13. An NNF-formula α is in LNNF, iff

- α is $\top$, $\bot$, or a literal l; or
- α is an $\wedge$-node $\beta_1 \wedge \cdots \wedge \beta_n$ where
  - for any non-trivial clause c, if $\alpha \models c$, then $\beta_j \models c$ for some $1 \leq j \leq n$; and
  - β_i is in LNNF for every $1 \leq i \leq n$; or
- α is an $\vee$-node $\beta_1 \vee \cdots \vee \beta_n$ where
  - for any non-trivial term t, if $t \models \alpha$, then $t \models \beta_j$ for some $1 \leq j \leq n$; and
  - β_i is in LNNF for every $1 \leq i \leq n$.

We use the following example to illustrate the distinction between LNNF and CE-LSNNF (resp. IM-LSNNF).

Example 7. Given the formula $\alpha = [((\neg x \vee y) \wedge (\neg y \vee z)) \vee \bot] \wedge (\neg x \vee z)$. It can be verified that α satisfies general CE-logical separability, but is not an LNNF-formula. Let $\beta = (\neg x \vee y) \wedge (\neg y \vee z)$. Each conjunct of β does not entail the clause $\neg x \vee z$, but β does. So β is not in LNNF. It follows that $\beta \vee \bot$ is not in LNNF. Neither does α. In a similar way, the two languages IM-LSNNF and LNNF can be distinguished by the NNF-formula $\bar{\alpha}$. □

From the above example, we can observe that LNNF incorporates not only a stronger constraint than CE-logical separability for $\wedge$-nodes but also its dual property for $\vee$-nodes. The direct consequences are (1) Both CE-LSNNF and IM-LSNNF subsume LNNF, and (2) LNNF supports polytime clausal entailment and term implication checking. CE-LSNNF is strictly more succinct than LNNF unless PH collapses, so is IM-LSNNF.

Proposition 13. LNNF $\not\leq_s^*$ CE-LSNNF *and* LNNF $\not\leq_s^*$ IM-LSNNF.

5. Model counting and counter-model counting

5.1. Model counting

We now turn to the query model counting that returns the number of models of a propositional formula. A number of key AI tasks can be reduced to the model counting problem, such as probabilistic inference [28] and constraint optimization [29].

We hereafter develop a unified model counting algorithm.

Definition 14. Let α be an NNF-formula, $\mathbf{Y} \supseteq \mathrm{Var}(\alpha)$ and $\mathbf{Z} = \mathbf{Y} \setminus \mathrm{Var}(\alpha)$. The procedure $\mathcal{CT}(\alpha, \mathbf{Y})$ is recursively defined as:

- $\mathcal{CT}(\bot, \mathbf{Y}) = 0$, $\mathcal{CT}(\top, \mathbf{Y}) = 2^{|\mathbf{Y}|}$ and $\mathcal{CT}(l, \mathbf{Y}) = 2^{|\mathbf{Y}|-1}$
- $\mathcal{CT}(\beta_1 \wedge \cdots \wedge \beta_n, \mathbf{Y}) = 2^{|\mathbf{Z}|} \cdot \prod_{i=1}^{n} \mathcal{CT}(\beta_i, \mathrm{Var}(\beta_i))$
- $\mathcal{CT}(\beta_1 \vee \cdots \vee \beta_n, \mathbf{Y}) = \sum_{i=1}^{n} \mathcal{CT}(\beta_i, \mathbf{Y})$

The above procedure works in a recursive way. The first item is for the base case. The numbers of models of $\top$, $\bot$ and a literal are $2^{|\mathbf{Y}|}$, 0 and $2^{|\mathbf{Y}|-1}$, respectively. In the induction case, the model counting problem of an $\wedge$-node (resp. $\vee$-node) is decomposed to that of its subformula. The number of models of an $\wedge$-node equals to the product of the number of models of its subformula β_i multiplying $2^{|\mathbf{Z}|}$. The number of models of an $\vee$-node equals to the sum of the $\mathbf{Y}$-models of its subformula β_i.

Definition 15. Let $\mathcal{L}$ be a language and f a procedure that takes an NNF-formula α and a set of variables $\mathbf{Y}$ s.t. $\mathbf{Y} \supseteq \mathrm{Var}(\alpha)$ as inputs, and that returns a natural number. We say

- f is CT-overapproximate for $\mathcal{L}$ iff for every $\mathcal{L}$-formula α and every set $\mathbf{Y}$ of variables, $\mathtt{CT}(\alpha, \mathbf{Y}) \leq f(\alpha, \mathbf{Y})$;
- f is CT-underapproximate for $\mathcal{L}$ iff for every $\mathcal{L}$-formula α and every set $\mathbf{Y}$ of variables, $\mathtt{CT}(\alpha, \mathbf{Y}) \geq f(\alpha, \mathbf{Y})$.

The procedure $\mathcal{CT}$ is overapproximate for any NNF-formula α with time linear in the size of α.

Theorem 13. *The procedure $\mathcal{CT}$ is CT-overapproximate for NNF with the time complexity $O(|\alpha|)$.*

However, the procedure $\mathcal{CT}$ may compute the incorrect answer for some NNF-formulas. We illustrate the reasons from the model-theoretic perspective.

We first consider the $\wedge$-node $\alpha = \beta_1 \wedge \cdots \wedge \beta_n$. Let Ω_i be the set of $\mathbf{Y}_i$-models of β_i where $\mathbf{Y}_i = \mathrm{Var}(\beta_i)$. The Cartesian product on $\Omega_1, \Omega_2, \cdots, \Omega_n$ may contain some pseudo-interpretations. The procedure $\mathcal{CT}$ simply returns the product of $|\Omega_1|, \cdots, |\Omega_n|$ and the number $2^{|\mathbf{Y}_0|}$. Therefore, $\mathcal{CT}$ computes an overapproximate result when the Cartesian product $\Omega_1 \times \cdots \times \Omega_n$ contains some pseudo-interpretations.

We now consider the $\vee$-node $\alpha = \beta_1 \vee \cdots \vee \beta_n$. Let Ω_i be the set of $\mathbf{Y}$-models of β_i. The procedure $\mathcal{CT}$ simply returns the sum of $|\Omega_i|$. Therefore, $\mathcal{CT}$ produces an overapproximate result since a $\mathbf{Y}$-interpretation may satisfy more than one disjunct of α.

Example 8 illustrates the above two cases in which procedure $\mathcal{CT}$ deduces the incorrect result.

Example 8. We first consider the formula $\alpha_1 = x \wedge [y \vee (x \wedge \neg y)]$ with two conjuncts $\beta_1 = x$ and $\beta_2 = y \vee (x \wedge \neg y)$. Let $\mathbf{Y} = \{x, y\}$, $\mathbf{Y}_i = \mathrm{Var}(\beta_i)$, and $\Omega_i = \mathrm{Mod}_{\mathbf{Y}_i}(\beta_i)$. We have that $\Omega_1 = \{\{x\}\}$ and $\Omega_2 = \{\{x, y\}, \{\neg x, y\}, \{x, \neg y\}\}$. Clearly, $\Omega_1 \times \Omega_2 = \{\{x, y\}, \{x, \neg x, y\}, \{x, \neg y\}\}$ contains a pseudo-interpretation $\{x, \neg x, y\}$. It follows that $\mathcal{CT}(\alpha_1, \mathbf{Y}) = |\Omega_1| \times |\Omega_2| = 3$. On the contrary, α_1 has only two $\mathbf{Y}$-models, hence $\mathtt{CT}(\alpha_1, \mathbf{Y}) = 2$.

We now turn to the formula $\alpha_2 = x \vee y$ with two disjuncts $\beta_1 = x$ and $\beta_2 = y$. Let $\Omega_i = \mathrm{Mod}_{\mathbf{Y}}(\beta_i)$. We have that $\Omega_1 = \{\{x, y\}, \{x, \neg y\}\}$ and $\Omega_2 = \{\{x, y\}, \{\neg x, y\}\}$. Clearly, β_1 and β_2 shares a $\mathbf{Y}$-model $\{x, y\}$. It follows that $\mathcal{CT}(\alpha_1, \mathbf{Y}) = |\Omega_1| + |\Omega_2| = 4$, which contradicts the fact that $\mathtt{CT}(\alpha_2, \mathbf{Y}) = 3$. □

In the following, we identify the conditions under which the procedure $\mathcal{CT}$ provides the exact counting of models. We say two formulas α and β *agree on common variables*, if for every variable $x \in \mathrm{Var}(\alpha) \cap \mathrm{Var}(\beta)$, we have $\alpha \models x$ and $\beta \models x$, or $\alpha \models \neg x$ and $\beta \models \neg x$.

Definition 16. An NNF-formula is CT-logically separable ($\mathtt{CT}_{ls}$) iff one of the following conditions hold

- α is a literal, $\top$ or $\bot$;
- α is an $\wedge$-node $\beta_1 \wedge \cdots \wedge \beta_n$ s.t.
  - if α is unsatisfiable, then β_i is unsatisfiable and $\mathtt{CT}_{ls}$ for some i;
  - if α is satisfiable, then every β_i is $\mathtt{CT}_{ls}$ and every two distinct β_i and β_j agree on common variables.
- α is an $\vee$-node $\beta_1 \vee \cdots \vee \beta_n$ s.t. $\beta_i \wedge \beta_j \models \bot$ for $i \neq j$ and every β_i is $\mathtt{CT}_{ls}$.

With the above constraints, no pseudo-interpretations exist in the Cartesian product on sets of models of conjuncts of $\wedge$-nodes. Neither common models in some distinct disjuncts of $\vee$-nodes does. We use CT-LSNNF to denote the subset of NNF in which any formula satisfies CT-logical separability property. The following example provides two formulas equivalent to the two formulas α_1 and α_2 in Example 8 that satisfy CT-logical separability property, respectively.

Example 9 *(Example 8 cont'd).* Consider two CT-LSNNF-formulas $\alpha'_1 = x \wedge [(x \wedge y) \vee (x \wedge \neg y)]$ and $\alpha'_2 = (x \wedge \neg y) \vee y$, which are equivalent to α_1 and α_2 respectively.

Let us examine the formula α'_1. Let $\beta_1 = x$ and $\beta_2 = (x \wedge y) \vee (x \wedge \neg y)$. Obviously, β_1 and β_2 share the variable x, and entail x. By Definition 16, it can be verified that α'_1 satisfies CT-logical separability. Let $\mathbf{Y} = \{x, y\}$ and $\mathbf{Y}_i = \mathrm{Var}(\beta_i)$. It follows that $\mathcal{CT}(\alpha'_1, \mathbf{Y}) = \mathcal{CT}(\beta_1, \mathbf{Y}_1) \times \mathcal{CT}(\beta_2, \mathbf{Y}_2) = |\{\{x\}\}| \times |\{\{x, y\}, \{x, \neg y\}\}| = 2$. As a result, $\mathcal{CT}(\alpha'_1, \mathbf{Y}) = \mathtt{CT}(\alpha'_1, \mathbf{Y})$.

As for the formula $\alpha'_2 = (x \wedge \neg y) \vee y$, we can observe that its two disjuncts $\beta_1 = x \wedge \neg y$ and $\beta_2 = y$ are logically contradictory, *i.e.*, $\beta_1 \wedge \beta_2 \models \bot$. Therefore, α'_2 is CT-logically separable. It follows that $\mathcal{CT}(\alpha'_2, \mathbf{Y}) = \mathcal{CT}(\beta_1, \mathbf{Y}) + \mathcal{CT}(\beta_2, \mathbf{Y}) = |\{\{x, \neg y\}\}| + |\{\{x, y\}, \{\neg x, y\}\}| = 3$, which coincides with $\mathtt{CT}(\alpha'_2, \mathbf{Y}) = 3$. □

As shown in Theorem 14, CT-LSNNF precisely capture the class of formulas that allow $\mathcal{CT}$ to produce a correct result.

Theorem 14. *A language $\mathcal{L}$ is* CT-LSNNF *iff the procedure $\mathcal{CT}$ is* CT*-overapproximate and* CT*-underapproximate for $\mathcal{L}$.*

Darwiche [16] proposed a model counting algorithm that is correct for sd-DNNF, that is, the subset of d-DNNF with the smoothness property requiring that $\mathrm{Var}(\beta_i) = \mathrm{Var}(\beta_j)$ for each $\vee$-node $\beta_1 \vee \cdots \vee \beta_n$ of an NNF-formula. By comparison, the $\mathcal{CT}$ algorithm has wider scope of application and is more efficient than Darwiche's algorithm since CT-LSNNF is a strict superset of sd-DNNF and transforming a CT-LSNNF-formula into an equivalent sd-DNNF-formula may cause a $O(|\mathbf{X}|)$ blowup in size where $|\mathbf{X}|$ is the number of variables.

We hereafter prove that d-DNNF and CT-LSNNF are polynomially equivalent.

Proposition 14. d-DNNF $\sim_p$ CT-LSNNF.

To the best of our knowledge, the maximal fragment of NNF supporting polytime model counting that appears in prior literature is d-DNNF. Now, the maximal fragment becomes CT-LSNNF since CT-LSNNF is a strict superset of d-DNNF.

A dual language, namely CCT-LSNNF, to CT-LSNNF, which is discussed in the next subsection, is not a subset of CT-LSNNF but supports polytime model counting.

Finally, we prove that every language $\mathcal{L}$ that is polynomially translatable into CT-LSNNF also satisfies **CT**. But the opposite direction remains open.

Theorem 15. *If a language $\mathcal{L}$ is polynomially translatable into* CT-LSNNF*, then $\mathcal{L}$ satisfies* **CT**.

We close this section by providing the upper bound of the CT-LSNNF membership problem.

Proposition 15. *Deciding if an* NNF*-formula is in* CT-LSNNF *is in Δ_2^P.*

5.2. Counter-model counting

Counter-model counting is the complement of model counting, which is to compute the number of counter-models of an NNF-formula.

Definition 17. Let α be an NNF-formula, $\mathbf{Y} \supseteq \mathrm{Var}(\alpha)$ and $\mathbf{Z} = \mathbf{Y} \setminus \mathrm{Var}(\alpha)$. The procedure $\mathcal{CCT}(\alpha, \mathbf{Y})$ is recursively defined as:

- $\mathcal{CCT}(\bot, \mathbf{Y}) = 2^{|\mathbf{Y}|}$, $\mathcal{CCT}(\top, \mathbf{Y}) = 0$ and $\mathcal{CCT}(l, \mathbf{Y}) = 2^{|\mathbf{Y}|-1}$
- $\mathcal{CCT}(\beta_1 \wedge \cdots \wedge \beta_n, \mathbf{Y}) = \sum_{i=1}^{n} \mathcal{CCT}(\beta_i, \mathbf{Y})$
- $\mathcal{CCT}(\beta_1 \vee \cdots \vee \beta_n, \mathbf{Y}) = 2^{|\mathbf{Z}|} \cdot \prod_{i=1}^{n} \mathcal{CCT}(\beta_i, \mathrm{Var}(\beta_i))$

Before analyzing the correctness of $\mathcal{CCT}$, we need the following definition:

Definition 18. Let $\mathcal{L}$ be a language and f a procedure that takes an NNF-formula α and a set of variables $\mathbf{Y}$ s.t. $\mathbf{Y} \supseteq \mathrm{Var}(\alpha)$ as inputs, and that returns a natural number. We say

- f is CCT-overapproximate for $\mathcal{L}$ iff for every $\mathcal{L}$-formula α and every set $\mathbf{Y}$ of variables, $\mathtt{CCT}(\alpha, \mathbf{Y}) \leq f(\alpha, \mathbf{Y})$;
- f is CCT-underapproximate for $\mathcal{L}$ iff for every $\mathcal{L}$-formula α and every set $\mathbf{Y}$ of variables, $\mathtt{CCT}(\alpha, \mathbf{Y}) \geq f(\alpha, \mathbf{Y})$.

The procedure $\mathcal{CCT}$ is overapproximate but not underapproximate for the language NNF.

Theorem 16. *The procedure $\mathcal{CCT}$ is* CCT*-overapproximate for* NNF *with the time complexity $O(|\alpha|)$.*

We hereafter offer the condition under which the procedure $\mathcal{CCT}$ provides the exact number of counter-models.

Definition 19. An NNF-formula is CCT-logically separable ($\mathtt{CCT}_{ls}$) iff one of the following conditions hold

- α is a literal, $\top$ or $\bot$;
- α is an $\wedge$-node $\beta_1 \wedge \cdots \wedge \beta_n$ s.t. $\top \models \beta_i \vee \beta_j$ for $i \neq j$ and every β_i is $\mathtt{CCT}_{ls}$;
- α is an $\vee$-node $\beta_1 \vee \cdots \vee \beta_n$ s.t.

- if α is valid, then β_i is valid and $\mathtt{CCT}_{ls}$ for some i;
- if α is falsifiable, then every β_i is $\mathtt{CCT}_{ls}$ and every two distinct $\bar{\beta}_i$ and $\bar{\beta}_j$ agree on common variables.

We use CCT-LSNNF to denote the subset of NNF in which any formula satisfies $\mathtt{CCT}_{ls}$. The following example illustrates the run of procedure $CC\mathcal{T}$ on a CCT-LSNNF-formula.

Example 10. Consider the formula $\alpha = \beta_1 \vee \beta_2$ where $\beta_1 = x$ and $\beta_2 = (x \vee y) \wedge (x \vee \neg y)$. By Definition 19, the literals x, y and $\neg y$ are $\mathtt{CCT}_{ls}$. Since x and y have no common variable, the falsifiable $\vee$-node $x \vee y$ is $\mathtt{CCT}_{ls}$. So is $x \vee \neg y$. This, together with the fact that $\top \models (x \vee y) \vee (x \vee \neg y)$, implies that β_2 is $\mathtt{CCT}_{ls}$. Let us now examine the CCT-logical separability of α. First, $\bar{\beta}_1 = \neg x$ and $\bar{\beta}_2 = (\neg x \wedge \neg y) \vee (\neg x \wedge y)$. Clearly, they both entails $\neg x$, and hence agree on common variables. It follows that α satisfies CCT-logical separability.

We now turn to the run of procedure $CC\mathcal{T}$ on the formula α. Let $\mathbf{Y} = \{x, y\}$ and $\mathbf{Y}_i = \mathrm{Var}(\beta_i)$. We have that $CC\mathcal{T}(\alpha, \mathbf{Y}) = CC\mathcal{T}(\beta_1, \mathbf{Y}_1) \times CC\mathcal{T}(\beta_2, \mathbf{Y}_2) = |\{\{\neg x\}\}| \times |\{\{\neg x, \neg y\}, \{\neg x, y\}\}| = 2$. It can be verified that α has exactly two $\mathbf{Y}$-counter-models: $\{\neg x, \neg y\}$ and $\{\neg x, y\}$. Clearly, $CC\mathcal{T}(\alpha, \mathbf{Y}) = \mathtt{CCT}(\alpha, \mathbf{Y})$. □

CCT-LSNNF precisely capture the class of formulas that allow $CC\mathcal{T}$ to produce a correct result.

Theorem 17. *A language $\mathcal{L}$ is* CCT-LSNNF *iff the procedure* CCT *is* CCT*-overapproximate and* CCT*-underapproximate.*

From Theorem 17, we can easily deduce that if a language $\mathcal{L}$ translatable into CCT-LSNNF in polynomial time, then $\mathcal{L}$ supports **CCT**. However, we cannot draw the consequence in the opposite direction yet.

Theorem 18. *If a language is polynomially translate into* CCT-LSNNF*, then $\mathcal{L}$ satisfies* **CCT**.

Although CCT-LSNNF is not a subset of CT-LSNNF, CCT-LSNNF also supports polytime model counting. Based on the algorithm $CC\mathcal{T}$, we can easily design a polytime algorithm that can get the exact model counts of α. Given a CCT-LSNNF-formula α, its model counts $\mathtt{CT}(\alpha, \mathbf{Y}) = 2^{|\mathbf{Y}|} - CC\mathcal{T}(\alpha, \mathbf{Y})$. Since procedure $CC\mathcal{T}$ produces exact number of the counter-model of α, $2^{|\mathbf{Y}|} - CC\mathcal{T}(\alpha, \mathbf{Y})$ is the correct answer for the models of α. Similarly, we obtain that CT-LSNNF satisfies polytime counter-model counting.

We close this section by providing the upper bound of the CCT-LSNNF membership problem.

Proposition 16. *Deciding if an* NNF*-formula is in* CCT-LSNNF *is in* Δ_2^{P}.

6. Model enumeration and counter-model enumeration

6.1. Model enumeration

We now turn to another query model enumeration, which aims to list all models of a given formula. We first show a recursive procedure $\mathcal{ME}(\alpha, \mathbf{Y})$ for model enumeration proposed in [13].

Definition 20. Let α be an NNF-formula, $\mathbf{Y} \supseteq \mathrm{Var}(\alpha)$ and $\mathbf{Z} = \mathbf{Y} \setminus \mathrm{Var}(\alpha)$. The procedure $\mathcal{ME}(\alpha, \mathbf{Y})$ is recursively defined as follows:

- $\mathcal{ME}(\bot, \mathbf{Y}) = \{\}$ and $\mathcal{ME}(\top, \mathbf{Y}) = \Omega_{\mathbf{Y}}$
- $\mathcal{ME}(x, \mathbf{Y}) = \{\{x\}\} \times \Omega_{\mathbf{Z}}$ and $\mathcal{ME}(\neg x, \mathbf{Y}) = \{\{\neg x\}\} \times \Omega_{\mathbf{Z}}$
- $\mathcal{ME}(\beta_1 \wedge \cdots \wedge \beta_n, \mathbf{Y}) = \mathcal{ME}(\beta_1, \mathrm{Var}(\beta_1)) \times \cdots \times \mathcal{ME}(\beta_n, \mathrm{Var}(\beta_n)) \times \Omega_{\mathbf{Z}}$
- $\mathcal{ME}(\beta_1 \vee \cdots \vee \beta_n, \mathbf{Y}) = \bigcup_{i=1}^{n} \mathcal{ME}(\beta_i, \mathbf{Y})$

The above procedure works in a recursive way. For simplicity, we only consider the case where $\mathbf{Y} = \mathrm{Var}(\alpha)$ as follows. As to the case $\mathbf{Y} \supset \mathrm{Var}(\alpha)$, we can obtain $\mathcal{ME}(\alpha, \mathbf{Y})$ by computing the Cartesian product of $\mathcal{ME}(\alpha, \mathrm{Var}(\alpha))$ and $\Omega_{\mathbf{Z}}$ due to the fact that α is independent of the variables in $\mathbf{Z}$. An empty set is the model of a Boolean constant $\bot$, and a set consisting of an empty set is the model of $\top$. In the case where α is a positive literal x (resp. negative literal $\neg x$), the procedure returns a set consisting of its model $\{\{x\}\}$ (resp. $\{\{\neg x\}\}$). As to the case $\wedge$-node, the procedure returns the Cartesian product of the models of its all conjuncts over $\mathrm{Var}(\beta_i)$. As to the case $\vee$-node, the procedure returns the union of the models of its disjuncts over $\mathbf{Y}$.

Definition 21. Let $\mathcal{L}$ be a language and f be a procedure that takes an NNF-formula α and a set of variables $\mathbf{Y} \supseteq \mathrm{Var}(\alpha)$ as input, and that returns a collection of sets of literals. We say

- f is ME-underapproximate for $\mathcal{L}$ iff for every $\mathcal{L}$-formula α and every set $\mathbf{Y}$ of variables, $f(\alpha, \mathbf{Y}) \subseteq \mathtt{ME}(\alpha, \mathbf{Y})$;
- f is ME-overapproximate for $\mathcal{L}$ iff for every $\mathcal{L}$-formula α and every set $\mathbf{Y}$ of variables, $\mathtt{ME}(\alpha, \mathbf{Y}) \subseteq f(\alpha, \mathbf{Y})$.

The procedure $\mathcal{ME}$ is an ME-overapproximate algorithm for NNF, which causes the exponential time complexity in the size of the input set of variables $\mathbf{Y}$ in the worst case.

Theorem 19. *The procedure $\mathcal{ME}$ is* ME*-overapproximate for* NNF *with the time complexity* $O(|\alpha| \cdot |\mathbf{Y}|^2 \cdot 3^{4\cdot|\mathbf{Y}|})$.

For arbitrary NNF-formula, the procedure $\mathcal{ME}$ is neither polytime nor ME-underapproximate. Similar to the case of model counting, we can attribute the cause to the fact that $\mathcal{ME}$ does not exclude the pseudo-interpretations occurring among some $\wedge$-nodes of an NNF-formula, which is enumerated by the procedure and hence derive an incorrect result. We illustrate this with the following example.

Example 11. Given a formula $\alpha = x \wedge (y \vee x)$ and $\mathbf{Y} = \{x, y\}$. By Definition 20, $\mathcal{ME}(y \vee x, \mathbf{Y}) = \{\{x, y\}, \{\neg x, y\}, \{x, \neg y\}\}$. It follows that $\mathcal{ME}(\alpha, \mathbf{Y}) = \mathcal{ME}(x, \{x\}) \times \mathcal{ME}(y \vee x, \mathbf{Y}) = \{\{x, y\}, \{x, \neg x, y\}, \{x, \neg y\}\}$. Apparently, $\{x, \neg x, y\}$ is a pseudo-interpretation.

Moreover, we now show that procedure $\mathcal{ME}$ is intractable for some NNF-formulas. Consider the formula $\beta = \bigvee_{i=1}^{n}(x_i \wedge \neg x_i)$, $\mathbf{Y} = \{x_1, \ldots, x_n\}$ and $\mathbf{Z}_i = \mathbf{Y} \setminus \{x_i\}$ for every $1 \leq i \leq n$. According to the procedure $\mathcal{ME}$, we get that $\mathcal{ME}(x_i, \{x_i\}) = \{\{x_i\}\}$ and $\mathcal{ME}(\neg x_i, \{x_i\}) = \{\{\neg x_i\}\}$. It follows that $\mathcal{ME}(x_i \wedge \neg x_i, \mathbf{Y}) = \{\{x_i, \neg x_i\}\} \times \Omega_i$, where $\Omega_i = \{\omega \mid \omega \text{ is a } \mathbf{Z}_i\text{-interpretation}\}$. Obviously, $\mathcal{ME}(\beta, \mathbf{Y})$ generates $n \times 2^{n-1}$ distinct elements, and hence the procedure $\mathcal{ME}$ takes exponential time in the size of β and the number of its models. On the contrary, β is unsatisfiable and contains no models. □

We provide a necessary and sufficient condition on NNF for which the $\mathcal{ME}$ procedure is overapproximate and underapproximate as follows.

Definition 22. Let α be an arbitrary NNF-formula, $\mathbf{Y} = \mathrm{Var}(\alpha)$ and $\mathbf{Y}_i = \mathrm{Var}(\beta_i)$. An NNF-formula α is ME-logically separable (ME_{ls}) iff it satisfies one of the following conditions:

- α is a literal, $\top$ or $\bot$;
- α is an $\wedge$-node $\beta_1 \wedge \cdots \wedge \beta_n$ s.t.
  - if α is unsatisfiable, then β_i is unsatisfiable and ME_{ls} for some i;
  - if α is satisfiable, then every β_i is ME_{ls} and every two distinct β_i and β_j agree on common variables.
- α is an $\vee$-node $\beta_1 \vee \cdots \vee \beta_n$, and each β_i is ME_{ls}.

According to the constraints above, any two distinct conjuncts β_i and β_j of an $\wedge$-node α ought to agree on common variables. Essentially, the ME-logical separability property forbids the occurrence of pseudo-interpretations in any $\wedge$-node, under which the correctness of the procedure $\mathcal{ME}$ is guaranteed. We use ME-LSNNF to denote the subset of NNF in which any formula satisfies ME-logical separability property. The following is an illustrating example of ME-logical separability property.

Example 12 *(Example 11 cont'd).* Recall the formula $\alpha = x \wedge (y \vee x)$, whose conjuncts $\beta_1 = x$ and $\beta_2 = y \vee x$ do not agree on common variable x. By constraining β_2 such that it entails x, we can get a ME-LSNNF-formula $\alpha' = \beta_1 \wedge \beta_2' = x \wedge [(x \wedge y) \vee x]$ equivalent to the original one α. In such case, $\mathcal{ME}(\beta_2', \mathbf{Y}) = \{\{x, y\}, \{x, \neg y\}\}$. It follows that $\mathcal{ME}(\alpha, \mathbf{Y}) = \mathcal{ME}(\beta_1', \{x\}) \times \mathcal{ME}(\beta_2', \mathbf{Y}) = \{\{x, y\}, \{x, \neg y\}\}$, containing no pseudo-interpretation. As a result, $\mathcal{ME}(\alpha', \mathbf{Y}) = \mathrm{ME}(\alpha', \mathbf{Y})$. □

Theorem 20. *A language $\mathcal{L}$ is* ME-LSNNF *iff the procedure $\mathcal{ME}$ is* ME*-underapproximate and* ME*-overapproximate for $\mathcal{L}$.*

Proposition 17. DNNF $\sim_p$ ME-LSNNF.

It turns out that ME-LSNNF and DNNF are polynomially translatable into each other, so they are equally succinct. However, we claim that none of PI and the four disjunctive closure based languages which also support **ME**, is polynomially translatable into ME-LSNNF. On the one hand, Bova et al. [30,31] has shown that there exists a class of monotone 2-CNF formulas (which is in KROM[$\vee$]) and negative 3-CNF formulas (which is in PI and the other three disjunctive closure based languages) has only exponential size DNNF representations. On the other hand, by the fact that all of them support **CD** and **CO**, and that **CD** and **CO** implies **ME**, we can conclude that they support polytime model enumeration checking [9,19].

Clearly, every language that is polynomially translatable into ME-LSNNF supports polytime model enumeration checking. However, it is not the case for the opposite direction.

Theorem 21. *If a language $\mathcal{L}$ is polynomially translatable into* ME-LSNNF*, then $\mathcal{L}$ satisfies* **ME**.

Finally, we provide the upper bound of checking membership in ME-LSNNF.

Proposition 18. *Deciding if an* NNF*-formula is in* ME-LSNNF *is in* Δ_2^P.

6.2. Counter-model enumeration

In contrast to model enumeration, counter-model enumeration is to list all the interpretations that falsifies a given formula. Clearly, a simple modification of the procedure $\mathcal{ME}$ is sufficient to obtain an algorithm for counter-model enumeration.

Definition 23. Let α be an NNF-formula, $\mathbf{Y} \supseteq \mathrm{Var}(\alpha)$ and $\mathbf{Z} = \mathbf{Y} \setminus \mathrm{Var}(\alpha)$. The procedure $\mathcal{CME}(\alpha, \mathbf{Y})$ is recursively defined as follows:

- $\mathcal{CME}(\top, \mathbf{Y}) = \{\}$ and $\mathcal{CME}(\bot, \mathbf{Y}) = \Omega_{\mathbf{Y}}$
- $\mathcal{CME}(x, \mathbf{Y}) = \{\{\neg x\}\} \times \Omega_{\mathbf{Z}}$ and $\mathcal{CME}(\neg x, \mathbf{Y}) = \{\{x\}\} \times \Omega_{\mathbf{Z}}$
- $\mathcal{CME}(\beta_1 \wedge \cdots \wedge \beta_n, \mathbf{Y}) = \bigcup_{i=1}^{n} \mathcal{CME}(\beta_i, \mathbf{Y})$
- $\mathcal{CME}(\beta_1 \vee \cdots \vee \beta_n, \mathbf{Y}) = \mathcal{CME}(\beta_1, \mathrm{Var}(\beta_1)) \times \cdots \times \mathcal{CME}(\beta_n, \mathrm{Var}(\beta_n)) \times \Omega_{\mathbf{Z}}$

Definition 24. Let $\mathcal{L}$ be a language and f be a procedure that takes an NNF-formula α and a set of variables $\mathbf{Y} \supseteq \mathrm{Var}(\alpha)$ as input, and that returns a collection of sets of literals. We say

- f is CME-underapproximate for $\mathcal{L}$ iff for every $\mathcal{L}$-formula α and every set $\mathbf{Y}$ of variables, $f(\alpha, \mathbf{Y}) \subseteq \mathrm{CME}(\alpha, \mathbf{Y})$;
- f is CME-overapproximate for $\mathcal{L}$ iff for every $\mathcal{L}$-formula α and every set $\mathbf{Y}$ of variables, $\mathrm{CME}(\alpha, \mathbf{Y}) \subseteq f(\alpha, \mathbf{Y})$.

Theorem 22. *The procedure $\mathcal{CME}$ is* CME*-overapproximate for* NNF *with the time complexity* $O(|\alpha| \cdot |\mathbf{Y}|^2 \cdot 3^{4 \cdot |\mathbf{Y}|})$.

In a similar case, we hereafter propose a dual property to ME-logical separability, which allows the procedure $\mathcal{CME}$ to arrive at the correct answer.

Definition 25. Let α be an arbitrary NNF-formula, $\mathbf{Y} = \mathrm{Var}(\alpha)$ and $\mathbf{Y}_i = \mathrm{Var}(\beta_i)$. An NNF-formula α is CME-logically separable (CME_{ls}) iff it satisfies one of the following conditions:

- α is a literal, $\top$ or $\bot$;
- α is an $\wedge$-node $\beta_1 \wedge \cdots \wedge \beta_n$, and each β_i is CME_{ls};
- α is an $\vee$-node $\beta_1 \vee \cdots \vee \beta_n$ s.t.
  - if α is valid, then β_i is valid and CME_{ls} for some i;
  - if α is falsifiable, then every β_i is CME_{ls} and every two distinct $\bar{\beta}_i$ and $\bar{\beta}_j$ agree on common variables.

We use CME-LSNNF to denote the subset of NNF in which any formula satisfies CME-logical separability property. We use the following example to elaborate the run of procedure $\mathcal{CME}$ on a CME-LSNNF-formula.

Example 13. Consider the formula $\alpha = \beta_1 \vee \beta_2 = (x \wedge \top) \vee [(y \vee x) \wedge \top]$. By Definition 25, the literals x, y and Boolean constant $\top$ are CME-logically separable. Since x and y share no variables, $y \vee x$ is also CME-logically separable. So, β_1 and β_2 are CME-logically separable. We have that $\bar{\beta}_1 = \neg x \vee \bot$ and $\bar{\beta}_2 = (\neg y \wedge \neg x) \vee \bot$. Since $\bar{\beta}_1 \models \neg x$ and $\bar{\beta}_2 \models \neg x$, α satisfies CME-logical separability.

Let $\mathbf{Y} = \{x, y\}$ and $\mathbf{Y}_i = \mathrm{Var}(\beta_i)$. We now examine the run of $\mathcal{CME}(\alpha, \mathbf{Y})$. It is easily verified that $\mathcal{CME}(\beta_2, \mathbf{Y}_2) = \mathcal{CME}(y \vee x, \mathbf{Y}_2) \cup \mathcal{CME}(\top, \mathbf{Y}_2) = \{\{\neg x, \neg y\}\} \cup \{\} = \{\{\neg x, \neg y\}\}$. We get that $\mathcal{CME}(\alpha, \mathbf{Y}) = \mathcal{CME}(\beta_1, \mathbf{Y}_1) \times \mathcal{CME}(\beta_2, \mathbf{Y}_2) = \{\{\neg x\}\} \times \{\{\neg x, \neg y\}\} = \{\{\neg x, \neg y\}\}$, which coincides with the fact that α has exactly one $\mathbf{Y}$-counter-model: $\{\neg x, \neg y\}$. □

We show that (1) CME_{ls} precisely captures the sufficient and necessary condition on NNF under which the procedure $\mathcal{CME}$ is both overapproximate and underapproximate; (2) every language polynomially translatable into CME-LSNNF satisfies **CME**, but the opposite case does not hold; (3) the complexity of the membership problem of CME-LSNNF is in Δ_2^P.

Theorem 23. *A language $\mathcal{L}$ is* CME-LSNNF *iff the procedure $\mathcal{CME}$ is* CME*-underapproximate and* CME*-overapproximate for $\mathcal{L}$.*

Theorem 24. *If a language $\mathcal{L}$ is polynomially translatable into* CME-LSNNF*, then $\mathcal{L}$ satisfies* **CME**.

Proposition 19. *Deciding if an* NNF*-formula is in* CME-LSNNF *is in* Δ_2^P.

We state that the dual languages ME-LSNNF and CME-LSNNF are incomparable in terms of succinctness. It can be verified that CME-LSNNF subsumes the language PI. This, together with the fact that ME-LSNNF $\not\leq_s$ PI (cf. Proposition 21), implies that ME-LSNNF is not at least as succinct as CME-LSNNF. Symmetrically, we can get that the opposite case holds.

Proposition 20. ME-LSNNF $\not\leq_s$ CME-LSNNF *and* CME-LSNNF $\not\leq_s$ ME-LSNNF.

7. Forgetting and ensuring

7.1. Forgetting

Forgetting, also typically referred to as *existential variable quantification* [32,33], is a crucial transformation in a number of AI applications, including conformant planning[1] [34,35], model-based diagnosis[2] [1] and model counting [36]. Informally, it aims at removing any information to some variables $\mathbf{Y}$ from a given propositional knowledge base α but maintains all information captured by α about the complement of $\mathbf{Y}$. The following is an illustrating example.

Example 14 *(Example 5 cont'd).* Recall the formula $\alpha = (\neg x \vee y) \wedge (\neg y \vee z)$, which states that x entails y and y entails z. Given that $\mathbf{Y} = \{y\}$. Since $\exists \mathbf{Y}.\alpha \equiv \alpha|_{y} \vee \alpha|_{\neg y} \equiv \neg x \vee z$, forgetting $\mathbf{Y}$ from α yields $\neg x \vee z$, stating that x entails z. Obviously, it is the strongest consequence of α that does not contain any variable in $\mathbf{Y}$, characterizing those complete information of α that is independent of the variable y.

We now focus on a recursive procedure for the forgetting transformation presented in [13].

Definition 26. Let α be an NNF-formula and $\mathbf{Y} \subseteq \mathbf{X}$. The procedure $\mathcal{FO}(\alpha, \mathbf{Y})$ is recursively defined as:

- $\mathcal{FO}(\bot, \mathbf{Y}) = \bot$ and $\mathcal{FO}(\top, \mathbf{Y}) = \top$
- $\mathcal{FO}(l, \mathbf{Y}) = \begin{cases} \top, & \text{if } \mathrm{Var}(l) \in \mathbf{Y} \\ l, & \text{otherwise} \end{cases}$
- $\mathcal{FO}(\beta_1 \wedge \cdots \wedge \beta_n, \mathbf{Y}) = \bigwedge_{i=1}^{n} \mathcal{FO}(\beta_i, \mathbf{Y})$
- $\mathcal{FO}(\beta_1 \vee \cdots \vee \beta_n, \mathbf{Y}) = \bigvee_{i=1}^{n} \mathcal{FO}(\beta_i, \mathbf{Y})$

In the base case where α is $\top$, $\bot$ or a $\overline{\mathbf{Y}}$-literal, the result of forgetting procedure is simply α itself. When α is a $\mathbf{Y}$-literal, the procedure returns $\top$. In the induction case, the procedure of forgetting $\mathbf{Y}$ from an $\wedge$-node (resp. $\vee$-node) is broken down into that of its subformula.

Definition 27. Let $\mathcal{L}$ be a language and f be a procedure that takes an NNF-formula α and a variable set $\mathbf{Y}$ as input, and that returns a modified NNF-formula mentioning no variable in $\mathbf{Y}$. We say

- f is FO-underapproximate for $\mathcal{L}$ iff for every $\mathcal{L}$-formula α and every $\mathbf{Y} \subseteq \mathbf{X}$, $\mathrm{FO}(\alpha, \mathbf{Y}) \models f(\alpha, \mathbf{Y})$;
- f is FO-overapproximate for $\mathcal{L}$ iff for every $\mathcal{L}$-formula α and every $\mathbf{Y} \subseteq \mathbf{X}$, $f(\alpha, \mathbf{Y}) \models \mathrm{FO}(\alpha, \mathbf{Y})$.

The procedure $\mathcal{FO}$ is FO-underapproximate for arbitrary NNF-formula.

Theorem 25. *The procedure $\mathcal{FO}$ is* FO*-underapproximate for* NNF *with the time complexity* $O(|\alpha| \cdot |\mathbf{Y}|)$.

Nevertheless, it is not the case that the procedure $\mathcal{FO}$ is FO-overapproximate when taking arbitrary NNF-formula as input. In other words, there exist a NNF-formula α and a set $\mathbf{Y}$ of variables such that the models of $\mathcal{FO}(\alpha, \mathbf{Y})$ contains counter-models of $\mathrm{FO}(\alpha, \mathbf{Y})$. Recall the example in Example 14, which is sufficient to illustrate this point.

Example 15 *(Example 14 cont'd).* Consider again $\alpha = (\neg x \vee y) \wedge (\neg y \vee z)$ and $\mathbf{Y} = \{y\}$. By Definition 26, $\mathcal{FO}(\neg x \vee y, \mathbf{Y}) = \neg x \vee \top$ and $\mathcal{FO}(\neg y \vee z, \mathbf{Y}) = \top \vee z$. We get that $\mathcal{FO}(\alpha, \mathbf{Y}) = (\neg x \vee \top) \wedge (\top \vee z)$, which is equivalent to $\top$. On the contrary, $\mathrm{FO}(\alpha, \mathbf{Y}) \equiv \neg x \vee z$. Obviously, in addition to all the models of $\mathrm{FO}(\alpha, \mathbf{Y})$, $\mathcal{FO}(\alpha, \mathbf{Y})$ also contains $\{x, y, \neg z\}$ and $\{x, \neg y, \neg z\}$ as their models. Therefore, $\mathcal{FO}(\alpha, \mathbf{Y}) \not\models \mathrm{FO}(\alpha, \mathbf{Y})$. □

The main reason is that the procedure $\mathcal{FO}$ just simply replaces every occurrence of each $\mathbf{Y}$-literal y and its complement $\neg y$ in two disjuncts $\neg x \vee y$ and $\neg y \vee z$ by $\top$, losing sight of the implicit logical implicate $\neg x \vee z$ over $\overline{\mathbf{Y}}$ occurring in α induced by variables in $\mathbf{Y}$. Based on this observation, we now turn to an equivalent CE-LSNNF-formula and reconsider the run of procedure $\mathcal{FO}$.

Example 16 *(Example 15 cont'd).* Recall (in Example 5) that $\alpha' = (\neg x \vee y) \wedge (\neg y \vee z) \wedge (\neg x \vee z)$ is a CE-LSNNF-formula equivalent to α by conjoining α with $\neg x \vee z$. We have that $\mathcal{FO}(\alpha', \mathbf{Y}) = (\neg x \vee \top) \wedge (\top \vee z) \wedge (\neg x \vee z)$, which is equivalent to $\mathrm{FO}(\alpha', \mathbf{Y}) \equiv \neg x \vee z$. The information $\neg x \vee z$ can be successfully preserved after eliminating variables of $\mathbf{Y}$ from α' via procedure $\mathcal{FO}$. □

[1] Conformant planning aims at finding a sequence of actions that achieves the goal under incomplete initial knowledge base.

[2] The goal of model-based diagnosis is to infer potential failure components from the knowledge base of system description, in light of an observation of the system.

Interestingly, CE-logical separability is exactly the sufficient and necessary condition on NNF under which the procedure $\mathcal{FO}$ is overapproximate and underapproximate.

Theorem 26. *A language $\mathcal{L}$ is* CE-LSNNF *iff the procedure $\mathcal{FO}$ is* FO*-overapproximate and* FO*-underapproximate for $\mathcal{L}$.*

With the constraint of CE-logical separability, any logical implicate over $\overline{\mathbf{Y}}$ of α occurs explicitly in some of its $\wedge$-nodes, which allows the procedure $\mathcal{FO}$ to eliminate any reference to $\mathbf{Y}$ without discarding any information captured by α about $\overline{\mathbf{Y}}$.

Theorem 27. *If a language $\mathcal{L}$ is polynomially equivalent to* CE-LSNNF*, then $\mathcal{L}$ satisfies* **FO**.

As the procedure $\mathcal{FO}$ preserves the CE-logical separability property, the language CE-LSNNF is closed under forgetting. Therefore, every language that is polynomially equivalent to CE-LSNNF supports polytime forgetting transformation property. We remark that compared to the query property, the transformation property not only needs a polytime procedure for the corresponding transformation task for the language $\mathcal{L}$, but also requires the output formula of the procedure to be in $\mathcal{L}$. Hence, unlike the Theorem 9 in Section 4.1, Theorem 27 requires polynomially equivalent rather than polynomially translatable. For example, the language IP, a strict subset of CE-LSNNF, does not support **FO**, due to the fact that forgetting some variables from certain IP-formulas derives an IP-representation with size exponential of the original one [9]. The opposite of Theorem 27 does not hold. It is worth to mention that the $\mathcal{FO}$ procedure for any IP-formula can generate a correct forgetting result in CE-LSNNF although it may not be an IP-formula. We find that all languages summarized in [9,19] permitting **FO**, including DNNF, DNF, PI, MODS and KROM[$\vee$], are polynomially translatable into CE-LSNNF. As far as we know, CE-LSNNF is the maximal fragment of NNF that supports **FO**.

7.2. Ensuring

Ensuring, also known as *universally variable quantification*, the dual transformation to forgetting, has an important application on conformant planning [34,35] and explainable AI [37]. In planning, contrary to the forgetting transformation which is often used to eliminate the current state variables when computing the symbolic representation of successor states, ensuring is usually applied when calculating the symbolic representation of predecessor states in the case of non-deterministic action effects. Intuitively, it aims at extracting the maximum amount of information about $\overline{\mathbf{Y}}$ from knowledge base α in a robust way, such that the extracted knowledge base always fulfills the original one α under uncertainty of assignment on variables in $\mathbf{Y}$. We illustrate this with the same example as in Example 14.

Example 17 *(Example 14 cont'd).* Recall that $\alpha = (\neg x \vee y) \wedge (\neg y \vee z)$ and $\mathbf{Y} = \{y\}$. Since $\forall \mathbf{Y}.\alpha \equiv (\alpha|y) \wedge (\alpha|\neg y) \equiv \neg x \wedge z$, ensuring $\mathbf{Y}$ from α yields $\neg x \wedge z$. Clearly, it is the weakest antecedent of α that is independent of the variable y. Thus, in order to achieve the specification α, a robust policy to y is to ensure that x is false and z is true. □

We obtain an overapproximate procedure $\mathcal{EN}$ for ensuring from the procedure $\mathcal{FO}$ via a slight modification: $\mathcal{EN}(l, \mathbf{Y}) = \bot$ when $\mathrm{Var}(l) \in \mathbf{Y}$.

Definition 28. Let $\mathcal{L}$ be a language and f be a procedure that takes an NNF-formula α and a variable set $\mathbf{Y}$ as input, and that returns a modified NNF-formula mentioning no variable in $\mathbf{Y}$. We say

- f is EN-underapproximate for $\mathcal{L}$ iff for every $\mathcal{L}$-formula α and every $\mathbf{Y} \subseteq \mathbf{X}$, $\mathrm{EN}(\alpha, \mathbf{Y}) \models f(\alpha, \mathbf{Y})$;
- f is EN-overapproximate for $\mathcal{L}$ iff for every $\mathcal{L}$-formula α and every $\mathbf{Y} \subseteq \mathbf{X}$, $f(\alpha, \mathbf{Y}) \models \mathrm{EN}(\alpha, \mathbf{Y})$.

The following example shows the run of procedure $\mathcal{EN}$.

Example 18 *(Example 17 cont'd).* It is easily verified that $\alpha = (\neg x \vee y) \wedge (\neg y \vee z)$ is in IM-LSNNF. The procedure $\mathcal{EN}$ works as follows. First, $\mathcal{EN}(\neg x \vee y, \mathbf{Y}) = \neg x \vee \bot$ and $\mathcal{EN}(\neg y \vee z, \mathbf{Y}) = \bot \vee z$. Then, the output of $\mathcal{EN}(\alpha, \mathbf{Y})$ is $(\neg x \vee \bot) \wedge (\bot \vee z)$, which is logically equivalent to $\neg x \wedge z$. Therefore, $\mathcal{EN}(\alpha, \mathbf{Y}) \equiv \mathrm{EN}(\alpha, \mathbf{Y})$ □

Finally, we show that (1) the procedure $\mathcal{EN}$ is EN-overapproximate for arbitrary NNF-formula; (2) IM-logical separability precisely captures the sufficient and necessary condition on NNF under which the procedure $\mathcal{EN}$ is both overapproximate and underapproximate; and (3) every language polynomially equivalent to IM-LSNNF supports polytime ensuring transformation property.

Theorem 28. *The procedure $\mathcal{EN}$ is* EN*-overapproximate for* NNF *with the time complexity $O(|\alpha| \cdot |\mathbf{Y}|)$.*

Theorem 29. *A language $\mathcal{L}$ is* IM-LSNNF *iff the procedure $\mathcal{EN}$ is* EN*-overapproximate and* EN*-underapproximate for $\mathcal{L}$.*

Theorem 30. *If a language $\mathcal{L}$ is polynomially equivalent to* IM-LSNNF*, then $\mathcal{L}$ satisfies* **EN**.

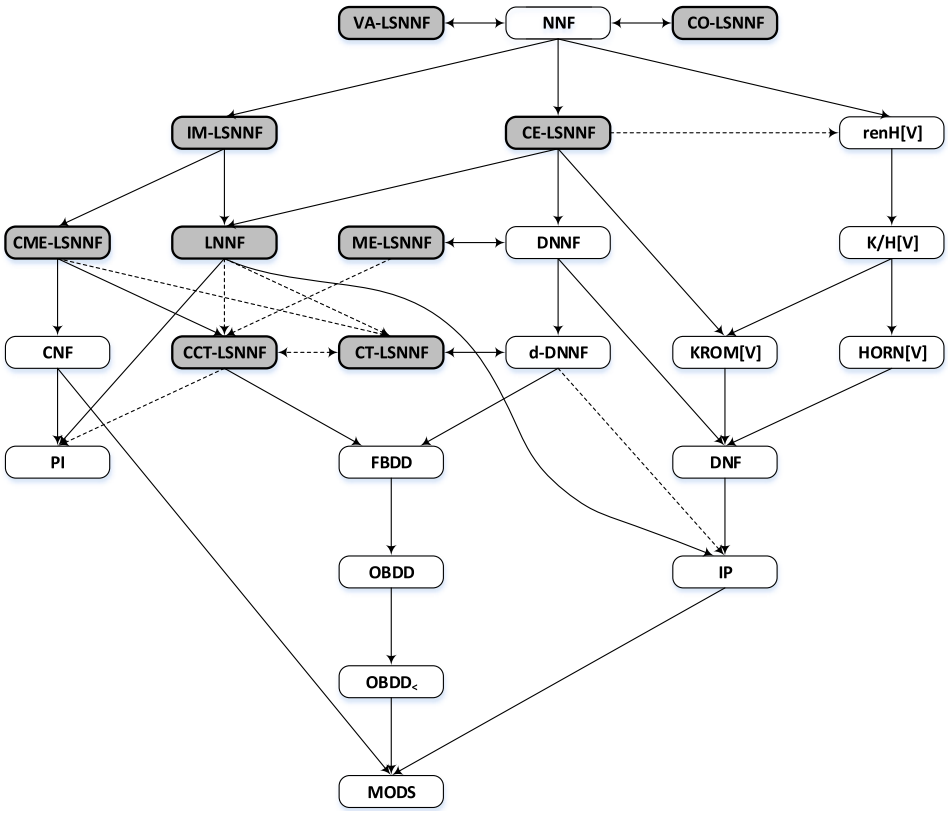


Fig. 1. Succinctness of various normal forms. An edge $\mathcal{L}_1 \rightarrow \mathcal{L}_2$ indicates $\mathcal{L}_1 <_s \mathcal{L}_2$ while an edge $\mathcal{L}_1 \leftrightarrow \mathcal{L}_2$ indicates $\mathcal{L}_1 \sim_s \mathcal{L}_2$. A dotted right arrow from $\mathcal{L}_1$ to $\mathcal{L}_2$ means $\mathcal{L}_2 \not\leq_s \mathcal{L}_1$ but whether $\mathcal{L}_1 \leq_s \mathcal{L}_2$ is unknown. A two-way dashed arrow between $\mathcal{L}_1$ and $\mathcal{L}_2$ indicates whether $\mathcal{L}_1 \leq_s \mathcal{L}_2$ and $\mathcal{L}_2 \leq_s \mathcal{L}_1$ remain unknown.

8. Succinctness, queries and transformations

As pointed out in [9], succinctness, queries and transformations are three key dimensions to consider when choosing an appropriate target language for application-specific problems. In this section, we discuss the relative succinctness of logical separability based languages, including CO-LSNNF, VA-LSNNF, CE-LSNNF, IM-LSNNF, LNNF, ME-LSNNF, CME-LSNNF, CT-LSNNF and CCT-LSNNF, compared to other languages considered in [9,19] in Fig. 1. We also investigate the supported polytime queries and transformations of logical separability based languages in Table 1.

Proposition 21. *The results in Fig. 1 and Table 1 hold.*

We can make several observations from Fig. 1 and Table 1 as follows. (1) There are four succinctness orderings of logical separability based languages. The first two orderings are CO-LSNNF $\sim_s$ VA-LSNNF $<_s$ CE-LSNNF $<_s$ ME-LSNNF $<_s$ CT-LSNNF and CO-LSNNF $\sim_s$ VA-LSNNF $<_s$ CE-LSNNF $<_s$ LNNF. Another two orderings are exactly their respective duals: CO-LSNNF $\sim_s$ VA-LSNNF $<_s$ IM-LSNNF $<_s$ CME-LSNNF $<_s$ CCT-LSNNF and CO-LSNNF $\sim_s$ VA-LSNNF $<_s$ IM-LSNNF $<_s$ LNNF. We know that CT-LSNNF (resp. CCT-LSNNF) is not at least as succinct as all of LNNF, IM-LSNNF and CME-LSNNF (resp. LNNF, CE-LSNNF and ME-LSNNF), but the opposite cases remain open. (2) Adding CO-logical separability to NNF gains the polytime satisfiability checking property without lowering down its succinctness hierarchy. But CO-LSNNF loses the transformation properties: **CD**, $\wedge$**BC**, $\wedge$**C** and $\neg$**C** which hold in NNF. It is worthy noting that CO-LSNNF is at least as succinct as any language that offers polytime satisfiability checking. Analogously, the addition of VA-logical separability on the NNF language contributes to tractability of validity checking, at the expense of losing several transformation properties: **CD**, $\vee$**BC**, $\vee$**C** and $\neg$**C**. VA-LSNNF is the most succinct language that supports polytime validity checking. (3) Besides KROM[$\vee$], ME-LSNNF and the subsets of DNNF, CE-LSNNF is another fragment of NNF supporting **CE**, **FO** and $\vee$**C**. Furthermore, CE-LSNNF is strictly more succinct than any languages supporting **CE** except renH[$\vee$], K/H[$\vee$], HORN[$\vee$] and CCT-LSNNF. (4) Although LNNF offers **VA** and **IM** compared to CE-LSNNF, but the former is not closed under bounded disjunction and is less succinct than the latter. (5) CT-LSNNF (resp. ME-LSNNF) is equally succinct to d-DNNF (resp. DNNF) and they support the same set of polytime query and transformation properties. (6) The logical separability based languages CE-LSNNF, LNNF, ME-LSNNF, CT-LSNNF and CCT-LSNNF satisfy polytime clausal entailment checking property, thus they qualify as target compilation languages. But none of them is closed under bounded conjunction.

Table 1
The supported polytime queries and transformations of the logical separability based languages. An occurrence of $\surd$ indicates that $\mathcal{L}$ supports the query (or transformation) property; ∘ means that it is not the case unless P = NP; and ? means that whether $\mathcal{L}$ supports this property is unknown.

$\mathcal{L}$	CO	VA	CE	IM	EQ	SE	CT	ME	CME	CCT
CO-LSNNF	$\surd$	∘	∘	∘	∘	∘	∘	∘	∘	∘
VA-LSNNF	∘	$\surd$	∘	∘	∘	∘	∘	∘	∘	∘
CE-LSNNF	$\surd$	∘	$\surd$	∘	∘	∘	∘	$\surd$	∘	∘
IM-LSNNF	∘	$\surd$	∘	$\surd$	∘	∘	∘	∘	$\surd$	∘
LNNF	$\surd$	$\surd$	$\surd$	$\surd$	∘	∘	∘	$\surd$	$\surd$	∘
CT-LSNNF	$\surd$	$\surd$	$\surd$	$\surd$	?	∘	$\surd$	$\surd$	$\surd$	$\surd$
CCT-LSNNF	$\surd$	$\surd$	$\surd$	$\surd$	?	∘	$\surd$	$\surd$	$\surd$	$\surd$
ME-LSNNF	$\surd$	∘	$\surd$	∘	∘	∘	∘	$\surd$	∘	∘
CME-LSNNF	∘	$\surd$	∘	$\surd$	∘	∘	∘	∘	$\surd$	∘

$\mathcal{L}$	CD	FO	SFO	EN	SEN	∧BC	∧C	∨BC	∨C	¬C
CO-LSNNF	∘	∘	$\surd$	∘	?	∘	∘	$\surd$	$\surd$	∘
VA-LSNNF	∘	∘	?	∘	$\surd$	$\surd$	$\surd$	∘	∘	∘
CE-LSNNF	$\surd$	$\surd$	$\surd$	∘	?	∘	∘	$\surd$	$\surd$	∘
IM-LSNNF	$\surd$	∘	?	$\surd$	$\surd$	$\surd$	$\surd$	∘	∘	∘
LNNF	$\surd$	?	?	?	?	∘	∘	∘	∘	$\surd$
CT-LSNNF	$\surd$	∘	∘	∘	∘	∘	∘	∘	∘	?
CCT-LSNNF	$\surd$	∘	∘	∘	∘	∘	∘	∘	∘	?
ME-LSNNF	$\surd$	$\surd$	$\surd$	∘	?	∘	∘	$\surd$	$\surd$	∘
CME-LSNNF	$\surd$	∘	?	$\surd$	$\surd$	$\surd$	$\surd$	∘	∘	∘

9. Related work

9.1. Extension of knowledge compilation map

The key idea underlying the concept of knowledge compilation is to convert knowledge bases into a suitable target language, so as to improve the efficiency of high computational complexity reasoning tasks. Aiming at offering a guideline for selecting a suitable target language for specific applications, Darwiche and Marquis [9] provided a knowledge compilation map that cross-ranks a dozen proportional languages in terms of their succinctness and tractability. Subsequently, a number of novel target languages have been proposed to extend the knowledge compilation map [20,38,21,22,11,39,40,19,23,24].

Wachter and Haenni [20] studied the knowledge compilation in the context of Propositional Directed Acyclic Graphs, a more general family of proportional languages that allowing negation symbol appears in the scope of arbitrary subformula rather than a variable. In order to gain the ∧**BC** property, Pipatsrisawat and Darwiche [21] introduced two novel target languages SDNNF and d-SDNNF by imposing the property of structured decomposability into DNNF and d-DNNF, respectively. Based on structured decomposability and strong determinism, Darwiche [11] presented a canonical language *sentential decision diagram* (SDD), which is later shown to be the same tractable as OBDD but strictly more succinct than it [41,42]. Subbarayan et al. [12] investigated the knowledge compilation properties of Tree-of-BDDs ($\text{ToOBDD}_<$), and showed that $\text{ToOBDD}_<$ is an interesting target language from the practical side. Subsequently, Fargier and Marquis [22] extended the notion of Tree-of-BDDs to additional propositional languages, and then developed a systematic study of the proposed languages in terms of both succinctness and tractability. In the two papers [38,19], the authors applied disjunctive closure principles to several NNF subsets, including $\text{OBDD}_<$, PI and four remarkable incomplete fragments of CNF, leading to a family of complete languages that are at least as succinct as them respectively. Besides, the authors extended the knowledge compilation map with the proposed disjunctive closure based languages, and showed that applying disjunctive closures preserves the properties of **CO**, **CD**, **CE** and **ME**. By exploiting symmetries to the languages FBDD and *decomposable decision diagrams* (DDG), Bart et al. [40] presented two symmetry-driven decision diagrams which have a higher degree of succinctness but preserve the property of **CT**. In the two papers [23,24], two classes of representations: *pseudo-Boolean constraints* (PBC) and *switch-list* (SL), are considered along the line of the knowledge compilation map in [9].

9.2. Knowledge compilation for specific reasoning tasks

In the area of knowledge compilation, it is also interesting to explore the tractability for a single specific reasoning task that plays a key role in some applications, including model counting [39,43] and boolean functional synthesis [39,44].

Knowledge compilation for model counting. The compilation-based approach proves to be a powerful scheme to address the model counting problem. In details, it first compiles the input formula into a target language tractable for model counting, and then invokes the corresponding polytime model counting algorithm to solve the model counting problem.

Most of the existing model counters based on this approach, including C2D [45], MINIC2D [46], DSHARP [47] and D4 [48], implement the conversion of the input formula into either d-SDNNF [21] or Decision-DNNF [49], both of which are strict subsets of the language d-DNNF. Koriche et al. [39] proposed a novel target language called *extended affine decision trees* (EADT) for model counting, and design an efficient model counter based on EADT compilation. They employ as the branching rule a generalized

Shannon expansion of the form $((x \leftrightarrow \beta) \wedge \alpha|_{x \leftarrow \beta}) \vee ((x \leftrightarrow \neg\beta) \wedge \alpha|_{x \leftarrow \neg\beta})$. This contributes to a more general determinism than the syntactic determinism imposed on the language Decision-DNNF. In the paper [43], literal equivalence, an effective preprocessing technique for both satisfiability and model counting [50,51], was used to design a target language *constrained conjunction & decision diagram* (CCDD) that is tractable for model counting. Specifically, the authors generalized the language Decision-DNNF by introducing a kind of conjunction node (called kernelized conjunction node) to capture the literal equivalence. Utilizing the literal equivalence detection and replacement, the subformulas during compilation can be simplified without affecting the number of their models.

Knowledge compilation for boolean functional synthesis. Boolean functional synthesis (BFnS) is key issue for a number of applications, including reactive strategy synthesis, automated program synthesis and so on [52]. Given a Boolean specification f that specifies a relation between inputs $\mathbf{Y}$ and outputs $\mathbf{Z}$, the goal of BFnS is to synthesize each $z \in \mathbf{Z}$ as a Skolem function α over $\mathbf{Y}$ such that the specification f is satisfied.

Akshay et al. [52] presented a two-phase algorithm BFSS for boolean functional synthesis, in which phase 1 generates approximate Skolem functions by invoking a forgetting operation the same as $\mathcal{FO}$, and phase 2 uses a CEGAR-based approach to produce correct Skolem functions. They identified an NNF fragment so-called wDNNF that allows phase 1 of BFSS to derive a correct result. The language satisfies *weak decomposability* property, which requires that for each $\wedge$-node $\alpha_1 \wedge \cdots \wedge \alpha_n$, there is no variable x and two distinct conjuncts α_i and α_j s.t. $x \in \text{Lit}(\alpha_i)$ and $\neg x \in \text{Lit}(\alpha_j)$. Subsequently, in paper [53], they proposed a language called SynNNF that precisely captures the specifications which admit correctly forgetting the output variables $\mathbf{Z}$ in accordance with a specific quantification order on $\mathbf{Z}$. By slightly weakening the property imposed on SynNNF, they offered a sufficient and necessary condition to guarantee the correctness of the first phase of BFSS in [52]. Furthermore, in the [44], they identified a normal form called SAUNF that precisely characterizes tractable boolean functional synthesis, that is, every language permits polytime synthesis if and only if it is polynomial translatable into SAUNF. The authors showed that SAUNF subsumes wDNNF and SynNNF, and is strictly more succinct than OBDD and DNNF. We remark that the language CE-LSNNF proposed in our paper is a strict subset of both SynNNF and SAUNF, hence it is also tractable for boolean functional synthesis. But whether SynNNF and SAUNF are strictly more succinct than CE-LSNNF remains unknown.

10. Conclusions and future work

Inspired by Levesque, we introduce the logical separability based properties for four queries and one transformation and their dual tasks respectively, establishing the connection between knowledge compilation and logical separability. We then identify a number of novel normal forms: CO-LSNNF, CE-LSNNF, CT-LSNNF, ME-LSNNF and their duals, which precisely capture the classes of formulas that permit procedures $\mathcal{CO}$, $\mathcal{CE}$, $\mathcal{CT}$ and $\mathcal{ME}$ always produce a correct answer of the corresponding tasks, respectively. We also extend the knowledge compilation map by investigating succinctness and tractability of the proposed logical separability based languages. We show that CO-LSNNF is the most succinct language permitting **CO** and particularly any propositional formula has a polysize equivalent CO-LSNNF formula. CE-LSNNF is as powerful as DNNF since they supports the same set of queries and transformations, yet the former is strictly more succinct than the later. Besides, CE-LSNNF is the maximal fragment of NNF that permits **FO** via procedure $\mathcal{FO}$. CT-LSNNF (resp. ME-LSNNF) is the same as succinct as d-DNNF (resp. DNNF) and offers the same tractability as d-DNNF (resp. DNNF).

There are several directions for future works. Firstly, the remaining open questions include: the unknown succinctness, supported queries and transformations which shown in Fig. 1 and Table 1 as well as the lower bound of the membership problem of some proposed languages. Secondly, it is interesting to develop a compilation method for CE-LSNNF, leading to build up a more efficient representation for two important AI problems: planning and model-based diagnosis. This is because that CE-LSNNF is the most succinct normal form, shown in Fig. 1, which can efficiently handle the four basic queries and transformations (clausal entailment, model enumeration, conditioning and forgetting) that are essential to planning and model-based diagnosis. Moreover, besides the queries considered in this paper, there are two remaining ones: equivalence and sentential entailment, both of which take as input two formulas. We would like to design a logical separability-based method to the two reasoning problems by induction on a pair of two formulas. Additionally, as for the four transformations (conditioning, conjunction, disjunction and negation), it is enough to derive a logically correct formulas for every normal form via simple syntactic operations. These operations however do not guarantee that the resulting formula satisfies the syntactic requirement of the normal form. We are interested in developing a uniform method to regain the normal form from the resulting formula so as to analyze the logical-separability property of the four transformations. Finally, it is also interesting to investigate the application of logical separability to some key reasoning tasks within the context of quantified propositional logic or other more expressive logics.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgements

This paper was supported by National Key R&D Program of China (No. 2022YFC3303603), National Natural Science Foundation of China (Nos. 62377028, 62276114, 62176103, 62077028 and 61906216), Guangdong-Macao Advanced Intelligent Computing Joint Laboratory (No. 2020B1212030003), the Guangdong Basic and Applied Basic Research Foundation (Nos. 2022A1515011309, 2021A1515011873, 2021B0101420003, 2021B1515120048 and 2020B0909030005), the Science and Technology Planning Project of Guangzhou (No. 202206030007), Key Laboratory of Smart Education of Guangdong Higher Education Institutes, Jinan University (No. 2022LSYS003), the Fundamental Research Funds for the Central Universities, JNU (No. 21623202), the XJTLU Research Development Fund (No. RDF-22-02-096) and the Guangxi Key Laboratory of Trusted Software Project (No. KX202314).

Appendix A. Supplemental proofs

Proof of Theorem 1. The procedure $\mathcal{CO}(\alpha)$ does a depth-first-traversal on the rooted DAG α. On each internal node, the conjunction or disjunction operation is performed n times where n is the number of its children. On each terminal node, it performs a constant amount of work. Therefore, the procedure $\mathcal{CO}(\alpha)$ takes $O(|\alpha|)$ time.

We prove that for any NNF formula α, if $\mathtt{CO}(\alpha)=1$, then $\mathcal{CO}(\alpha)=1$ by induction on α. Suppose that α is satisfiable. It remains to verify that $\mathcal{CO}(\alpha)=1$.

- It directly follows from Definition 2 that $\mathcal{CO}(x)=\mathcal{CO}(\neg x)=\mathcal{CO}(\top)=1$.
- α is an $\wedge$-node $\beta_1\wedge\cdots\wedge\beta_n$: we get that each β_i is satisfiable as α is satisfiable. By the induction hypothesis, we have $\mathcal{CO}(\beta_i)=1$ for each i. Therefore, it holds that $\mathcal{CO}(\alpha)=1$.
- α is an $\vee$-node $\beta_1\vee\cdots\vee\beta_n$: we get that β_i is satisfiable for some i as α is satisfiable. By the induction hypothesis, we have $\mathcal{CO}(\beta_i)=1$ for some i. Therefore, it holds that $\mathcal{CO}(\alpha)=1$. □

Proof of Theorem 2. ($\Rightarrow$): By Theorem 1 and the fact that CO-LSNNF is a subset of NNF, we get that the procedure $\mathcal{CO}$ is complete for CO-LSNNF. It remains to verify the soundness property of the procedure $\mathcal{CO}$ for CO-LSNNF, that is, for any CO-LSNNF formula α, if $\mathcal{CO}(\alpha)=1$, then $\mathtt{CO}(\alpha)=1$. We prove by contraposition, and assume that α is unsatisfiable. We prove that $\mathcal{CO}(\alpha)=0$ by induction on α.

- It directly follows from Definition 2 that $\mathcal{CO}(\bot)=0$.
- α is an $\wedge$-node $\beta_1\wedge\cdots\wedge\beta_n$: By CO-logical separability property of α, there is some β_i that is unsatisfiable and $\mathtt{CO}_{ls}$. By the induction hypothesis, we have $\mathcal{CO}(\beta_i)=0$. It follows from Definition 2 that $\mathcal{CO}(\alpha)=0$.
- α is an $\vee$-node $\beta_1\vee\cdots\vee\beta_n$: By CO-logical separability property of α, every β_i is unsatisfiable and $\mathtt{CO}_{ls}$. By the induction hypothesis, we have $\mathcal{CO}(\beta_i)=0$ for each $1\le i\le n$. It follows from Definition 2 that $\mathcal{CO}(\alpha)=0$.

($\Leftarrow$): Suppose that the language $\mathcal{L}$ is such that the procedure $\mathcal{CO}$ is complete and sound w.r.t. CO, that is, for any $\mathcal{L}$-formula α, $\mathtt{CO}(\alpha)=1$ iff $\mathcal{CO}(\alpha)=1$. We will prove that every $\mathcal{L}$-formula α is CO-logically separable by induction on α.

- By Definition 4, the Boolean constants $\top$ and $\bot$ and all literals are in CO-LSNNF.
- α is an $\wedge$-node $\beta_1\wedge\cdots\wedge\beta_n$. By Definition 4, if α is satisfiable, then $\alpha\in$ CO-LSNNF. We now assume that α is unsatisfiable (*i.e.* $\mathtt{CO}(\alpha)=0$). By the assumption that $\mathtt{CO}(\alpha)=1$ iff $\mathcal{CO}(\alpha)=1$, we get that $\mathcal{CO}(\alpha)=0$. According to the procedure $\mathcal{CO}$, we get that $\mathcal{CO}(\beta_i)=0$ for some i. Since the procedure $\mathcal{CO}$ is complete, β_i is unsatisfiable, and hence $\mathtt{CO}(\beta_i)=0$. It follows that $\beta_i\in\mathcal{L}$. By the induction hypothesis, β_i is CO-logically separable. By Definition 4, α is also CO-logically separable.
- α is an $\vee$-node $\beta_1\vee\cdots\vee\beta_n$: The proof is similar to the above case except that we take all β_i's into consideration. □

Proof of Proposition 1. By Definition 4, any satisfiable formula is a CO-LSNNF formula. So it is remains to verify every unsatisfiable DNNF and PI formula is in CO-LSNNF.

We first consider DNNF. We prove every unsatisfiable DNNF-formula is CO-logically separable by induction on α.

- α is of form $\top$, $\bot$ or a literal. By Definition 4, α is CO-logically separable.
- α is an $\wedge$-node $\beta_1\wedge\cdots\wedge\beta_n$. It follows that every β_i satisfies decomposability. By the property of decomposability, we get that some β_i is unsatisfiable. By the induction hypothesis, β_i is CO-logically separable. By Definition 4, α is CO-logically separable.
- α is an $\vee$-node $\beta_1\vee\cdots\vee\beta_n$. It follows that every β_i satisfies decomposability. Since α is unsatisfiable, every β_i is unsatisfiable. By the induction hypothesis, every β_i is CO-logically separable. By Definition 4, α is CO-logically separable.

We now consider PI. An unsatisfiable PI-formula α entails $\bot$. The Boolean constant $\bot$ is a disjunction of an empty set of literals, and hence is a clause. It follows that $\bot$ is an implicate of α. By the definition of PI, there is no implicate $c'\neq c$ s.t. $\alpha\models c'$ and $c'\models c$. Since α must be $\bot$, and is CO-logically separable. □

Proof of Theorem 3. ($\Rightarrow$): Assume that $\mathcal{L}$ supports polytime satisfiability checking. We design an algorithm transforming any $\mathcal{L}$-formula α into an equivalent one that is CO-logically separable. If it is satisfiable, then it is CO-logically separable; otherwise, the

algorithm returns $\bot$ which is CO-logically separable. Since the satisfiability of $\mathcal{L}$-formula α can be achieved in time polynomial in $|\alpha|$, the above algorithm is also polytime.

($\Leftarrow$): Let f be the polytime algorithm that transforms any $\mathcal{L}$-formula α into an equivalent one that is CO-logically separable. By the completeness and soundness of the procedure $\mathcal{CO}$ for CO-LSNNF, α is satisfiable iff $\mathcal{CO}(f(\alpha)) = 1$. This, together with the fact that f and $\mathcal{CO}$ are polytime, imply that $\mathcal{L}$ satisfies **CO**. □

Proof of Proposition 2. (NNF $\sim_s$ CO-LSNNF): Clearly, NNF $\leq_s$ CO-LSNNF as CO-LSNNF is a subset of NNF. The fact that CO-LSNNF $\leq_s$ NNF also holds as every satisfiable NNF-formula is a CO-LSNNF-formula and every unsatisfiable NNF formula is equivalent to $\bot$, which is a CO-LSNNF-formula.

(NNF $\leq_p$ CO-LSNNF): It directly follows from the fact that CO-LSNNF is a subset of NNF.

(CO-LSNNF $\not\leq_p^*$ NNF): We assume that CO-LSNNF $\leq_p$ NNF. There is a deterministic algorithm f that transforms any NNF-formula α into an equivalent CO-LSNNF-formula $f(\alpha)$. The satisfiability of CO-LSNNF can be tractably solved by the procedure $\mathcal{CO}$. So is NNF. This leads to the statement that NP = P, which implies that PH collapses. □

Proof of Proposition 3. (Lower bound): Let IsCOLSNNF be the procedure that takes as input an NNF-formula α, and returns 1 if $\alpha \in$ CO-LSNNF; 0, otherwise. Based on the procedure IsCOLSNNF, we design a procedure f that takes as input an NNF-formula α, and returns 1 if α is satisfiable; 0, otherwise.

$$f(\alpha) = \begin{cases} 1, & \text{if IsCOLSNNF}(\alpha) = 1 \text{ and } \mathcal{CO}(\alpha) = 1; \\ 0, & \text{otherwise.} \end{cases}$$

By Definition 4 and Theorem 2, we get that for any NNF formula α, α is satisfiable iff α is an CO-LSNNF-formula and $\mathcal{CO}(\alpha)$ returns 1. Due to the fact that the satisfiability problem of any NNF-formula is NP-complete, we can draw a conclusion that deciding if an NNF-formula is in CO-LSNNF is NP-hard.

(Upper bound): Based on the satisfiability problem CO and the procedure $\mathcal{CO}$ (Algorithm 2), the procedure IsCOLSNNF(α) is defined as follows:

$$\text{IsCOLSNNF}(\alpha) = \begin{cases} 1, & \text{if CO}(\alpha) = 1 \text{ or } \mathcal{CO}(\alpha) = 0; \\ 0, & \text{otherwise.} \end{cases}$$

By Definition 4 and Theorem 2, we get that for any NNF formula α, α is an CO-LSNNF-formula iff α is satisfiable or $\mathcal{CO}(\alpha)$ returns 0. Since the satisfiability problem is an NP decision problem and the procedure IsCOLSNNF calls the oracle CO once, deciding if an NNF-formula is in CO-LSNNF is in NP. □

Proof of Theorem 4. The proof is similar to that of Theorem 1. □

Proof of Theorem 5. The proof is similar to that of Theorem 2. □

Proof of Theorem 6. The proof is similar to that of Theorem 3. □

Proof of Proposition 4. The proof is similar to that of Proposition 2. □

Proof of Proposition 5. The proof is similar to that of Proposition 3.

(Lower bound): It holds that for any NNF formula α, α is falsifiable iff α is in VA-LSNNF and $\mathcal{VA}(\alpha) = 0$. Due to the fact that the deciding if an NNF-formula is falsifiable is NP-complete, we can draw a conclusion that checking membership in VA-LSNNF is NP-hard.

(Upper bound): It is easy to verify that for any NNF formula α, α is in VA-LSNNF iff α is falsifiable or $\mathcal{VA}(\alpha) = 1$. So the membership problem of VA-LSNNF is in NP. □

Proof of Proposition 6. ($\Rightarrow$): We prove that for any falsifiable non-trivial clause c, if $\mathcal{CE}(\alpha, c) = 1$ then $\mathcal{CO}(\alpha|t) = 0$, where t is a non-trivial term equivalent to $\neg c$. Assume that $c \not\equiv \top$ and that $\mathcal{CE}(\alpha, c) = 1$. We prove that $\mathcal{CO}(\alpha|t) = 0$ by induction on α.

- α is $\top$, $\bot$ or a literal l: By the assumption and Definition 7, we have that $\mathcal{CE}(\bot, \bot) = \mathcal{CE}(\bot, l_1 \vee \cdots \vee l_n) = 1$ and that $\mathcal{CE}(l, l_1 \vee \cdots \vee l_n) = 1$ when $l = l_i$ for some i. By the definition of conditioning, it holds that $\bot|\top = \bot|(\neg l_1 \wedge \cdots \wedge \neg l_n) = \bot$, and that $l|(\neg l_1 \wedge \cdots \wedge \neg l_n) = \bot$ when $l = l_i$ for some i. By Definition 2, we have that $\mathcal{CO}(\bot|\top) = \mathcal{CO}(\bot|(\neg l_1 \wedge \cdots \wedge \neg l_n)) = 0$, and that $\mathcal{CO}(l|(\neg l_1 \wedge \cdots \wedge \neg l_n)) = 0$ when $l = l_i$ for some i.
- α is an $\wedge$-node $\beta_1 \wedge \cdots \wedge \beta_n$: By the assumption and Definition 7, we have that $\mathcal{CE}(\beta_i, c) = 1$ for some i. By the induction hypothesis, we have $\mathcal{CO}(\beta_i|t) = 0$. According to the definition of conditioning, $\alpha|t = \beta_1|t \wedge \cdots \wedge \beta_n|t$ holds. By Definition 2, it follows that $\mathcal{CO}(\alpha|t) = 0$.
- α is an $\vee$-node $\beta_1 \vee \cdots \vee \beta_n$: By the assumption and Definition 7, we have that $\mathcal{CE}(\beta_i, c) = 1$ for every i. By the induction hypothesis, we have $\mathcal{CO}(\beta_i|t) = 0$. According to the definition of conditioning, $\alpha|t = \beta_1|t \vee \cdots \vee \beta_n|t$ holds. By Definition 2, it follows that $\mathcal{CO}(\alpha|t) = 0$.

($\Leftarrow$): It remains to prove that for any falsifiable non-trivial clause c, if $\mathcal{CO}(\alpha|t) = 0$ then $\mathcal{CE}(\alpha, c) = 1$, where t is a non-trivial term equivalent to $\neg c$. We prove by contraposition and assume that $c \not\equiv \top$ and that $\mathcal{CE}(\alpha, c) = 0$. We prove that $\mathcal{CO}(\alpha|t) = 1$ by induction on α.

- α is $\top$, $\bot$ or a literal l: By the assumption and Definition 7, we have that $\mathcal{CE}(\top, \bot) = \mathcal{CE}(\top, l_1 \vee \cdots \vee l_n) = \mathcal{CE}(l, \bot) = 0$ and that $\mathcal{CE}(l, l_1 \vee \cdots \vee l_n) = 0$ when $l \neq l_i$ for every i. By the definition of conditioning, it holds that $\top|\top = \top|(\neg l_1 \wedge \cdots \wedge \neg l_n) = \top$, $l|\top = l$ and that $l|(\neg l_1 \wedge \cdots \wedge \neg l_n) = \top$ when $l = \neg l_i$ for some i, and that $l|(\neg l_1 \wedge \cdots \wedge \neg l_n) = l$ when both $l \neq l_i$ and $l \neq \neg l_i$ for every i. By Definition 2, we have that $\mathcal{CO}(\top|\top) = \mathcal{CO}(\top|(\neg l_1 \wedge \cdots \wedge \neg l_n)) = 1$, $\mathcal{CO}(l|\top) = 1$, and that $\mathcal{CO}(l|(\neg l_1 \wedge \cdots \wedge \neg l_n)) = 1$ when $l \neq l_i$ for every i.
- α is an $\wedge$-node $\beta_1 \wedge \cdots \wedge \beta_n$: By the assumption and Definition 7, we have that $\mathcal{CE}(\beta_i, c) = 0$ for every i. By the induction hypothesis, we have $\mathcal{CO}(\beta_i|t) = 1$. According to the definition of conditioning, $\alpha|t = \beta_1|t \wedge \cdots \wedge \beta_n|t$ holds. By Definition 2, it follows that $\mathcal{CO}(\alpha|t) = 1$.
- α is an $\vee$-node $\beta_1 \vee \cdots \vee \beta_n$: By the assumption and Definition 7, we have that $\mathcal{CE}(\beta_i, c) = 0$ for some i. By the induction hypothesis, we have $\mathcal{CO}(\beta_i|t) = 1$. According to the definition of conditioning, $\alpha|t = \beta_1|t \vee \cdots \vee \beta_n|t$ holds. By Definition 2, it follows that $\mathcal{CO}(\alpha|t) = 1$. □

Proof of Theorem 7. The analysis of time complexity of the procedure $\mathcal{CE}$ is similar to that of $\mathcal{CO}$ except that it performs $O(|c|)$ work on each terminal node where c is the clause. Therefore, the procedure $\mathcal{CE}(\alpha, c)$ takes $O(|\alpha| \cdot |c|)$ time.

We prove that for any NNF formula α, $\mathcal{CE}(\alpha, c) = 1$ implies that $\mathtt{CE}(\alpha, c) = 1$ by induction on α. Suppose that $\mathcal{CE}(\alpha, c) = 1$. It remains to verify that $\mathtt{CE}(\alpha, c) = 1$.

- It directly follows from Definition 7 that $\mathtt{CE}(\top, \top) = \mathtt{CE}(\bot, c) = \mathtt{CE}(l, \top) = 1$, and that $\mathtt{CE}(l, l_1 \vee \cdots \vee l_m) = 1$ when $l = l_i$ for some i.
- α is an $\wedge$-node $\beta_1 \wedge \cdots \wedge \beta_n$: we get that $\mathcal{CE}(\beta_i, c) = 1$ for some i. By the induction hypothesis, we have $\mathtt{CE}(\beta_i, c) = 1$ for some i. Therefore, it holds that $\mathtt{CE}(\alpha, c) = 1$.
- α is an $\vee$-node $\beta_1 \vee \cdots \vee \beta_n$: we get that $\mathcal{CE}(\beta_i, c) = 1$ for every i. By the induction hypothesis, we have $\mathtt{CE}(\beta_i, c) = 1$ for every i. Therefore, it holds that $\mathtt{CE}(\alpha, c) = 1$. □

Proof of Theorem 8. ($\Rightarrow$): By Theorem 7 and the fact that CE-LSNNF is a subset of NNF, we get that the procedure $\mathcal{CE}$ is sound w.r.t. CE for CE-LSNNF. It remains to verify the completeness property of the procedure $\mathcal{CE}$ for CE-LSNNF, that is, for every CE-LSNNF formula α and every non-trivial clause c, $\mathtt{CE}(\alpha, c) = 1$ implies that $\mathcal{CE}(\alpha, c) = 1$. We assume that $\alpha \models c$ and prove by induction on α.

- It directly follows from Definition 7 that $\mathcal{CE}(\top, \top) = \mathcal{CE}(\bot, c) = \mathcal{CE}(l, \top) = 1$, and that $\mathcal{CE}(l, l_1 \vee \cdots \vee l_m) = 1$ when $l = l_i$ for some i.
- α is an $\wedge$-node $\beta_1 \wedge \cdots \wedge \beta_n$: By CE-logical separability property, we have that $\beta_i \models c$ and β_i is $\mathtt{CE}_{ls}$ w.r.t. c for some i. By the induction hypothesis, it holds that $\mathcal{CE}(\beta_i, c) = 1$. It follows from the procedure $\mathcal{CE}$ (Definition 7) that $\mathcal{CE}(\alpha, c) = 1$.
- α is an $\vee$-node $\beta_1 \vee \cdots \vee \beta_n$: By CE-logical separability property, we have that every β_i is $\mathtt{CE}_{ls}$ w.r.t. c and $\beta_i \models c$. By the induction hypothesis, it holds that $\mathcal{CE}(\beta_i, c) = 1$. It follows from the procedure $\mathcal{CE}$ that $\mathcal{CE}(\alpha, c) = 1$.

($\Leftarrow$): Suppose that the language $\mathcal{L}$ is such that the procedure $\mathcal{CE}$ is complete and sound w.r.t. clausal entailment, that is, for every $\mathcal{L}$-formula α and every non-trivial clause c, we have $\mathtt{CE}(\alpha, c) = 1$ iff $\mathcal{CE}(\alpha, c) = 1$. Let c be a non-trivial clause. We will prove that every $\mathcal{L}$-formula α is CE-logically separable w.r.t. c by induction on α.

- By Definition 9, the Boolean constants $\top$ and $\bot$ and any literal is in CE-LSNNF.
- α is an $\wedge$-node $\beta_1 \wedge \cdots \wedge \beta_n$: By Definition 9, if $\alpha \not\models c$, then α is $\mathtt{CE}_{ls}$ w.r.t. c. We now assume that $\alpha \models c$ (*i.e.* $\mathtt{CE}(\alpha, c) = 1$). By the assumption that $\mathtt{CE}(\alpha, c) = 1$ iff $\mathcal{CE}(\alpha, c) = 1$, we get that $\mathcal{CE}(\alpha, c) = 1$. According to the procedure $\mathcal{CE}$, we get that $\mathcal{CE}(\beta_i, c) = 1$ for some i. Since the procedure $\mathcal{CE}$ is sound for any NNF-formula, we have that $\mathtt{CE}(\beta_i, c) = 1$. It follows that $\beta_i \in \mathcal{L}$. By the induction hypothesis, β_i is CE-logically separable w.r.t. c. By Definition 9, α is $\mathtt{CE}_{ls}$ w.r.t. c.
- α is an $\vee$-node $\beta_1 \vee \cdots \vee \beta_n$: The proof is similar to the above case except that we take all β_i's into consideration. □

Proof of Proposition 7. We first prove that DNNF is a subset of CE-LSNNF. Let $\alpha \in$ DNNF and t a non-trivial term. Since for any $\wedge$-node $\beta_1 \wedge \cdots \wedge \beta_n$ of α, we get that $\mathrm{Var}(\beta_i|t) \subseteq \mathrm{Var}(\beta_i)$ for every i. So $\alpha|t \in$ DNNF. By Proposition 1, α satisfies CO-logical separability. By Proposition 9, $\alpha \in$ CE-LSNNF.

We now prove that PI is a subset of CE-LSNNF. We need to prove that if a NNF-formula α is in PI, then it is in CE-LSNNF. We prove by induction on α.

- By Definition 9, the Boolean constants $\top$ and $\bot$ and all literals l are CE-LSNNF.
- α is an $\wedge$-node $\beta_1 \wedge \cdots \wedge \beta_n$: By the definition of PI, it holds that for any non-trivial clause c, if $\alpha \models c$, then $\beta_i \models c$ and β_i is in PI for some i. By the induction hypothesis, we have β_i is $\mathtt{CE}_{ls}$. It follows from Definition 9 that α is $\mathtt{CE}_{ls}$ w.r.t. any non-trivial clause c.
- α is an $\vee$-node $\beta_1 \vee \cdots \vee \beta_n$: By the definition of PI, it holds that each β_i is in PI, since β_i is $\top$, $\bot$ or a literal l. By the induction hypothesis, we have β_i is $\mathtt{CE}_{ls}$. It follows from Definition 9 that α is $\mathtt{CE}_{ls}$ w.r.t. any non-trivial clause c. □

Proof of Theorem 9. Let f be the polytime algorithm that transforms any $\mathcal{L}$-formula α into an equivalent CE-LSNNF-formula. By the completeness and soundness of the procedure $\mathcal{CE}$ for CE-LSNNF, $\alpha \models c$ iff $\mathcal{CE}(f(\alpha), c) = 1$. This, together with the fact that f and $\mathcal{CE}$ are polytime, imply that $\mathcal{L}$ satisfies **CE**. □

Proof of Proposition 8. (CO-LSNNF $\leq_s$ CE-LSNNF): By CE-logical separability (Definition 9) and CO-logical separability property (Definition 4), we get that CE-LSNNF $\subseteq$ CO-LSNNF since every CO-LSNNF-formula is a formula that is CE-logically separable w.r.t. $\bot$. Therefore, CO-LSNNF $\leq_s$ CE-LSNNF.
(CE-LSNNF $\not\leq_s^*$ CO-LSNNF): By Theorem 31 and the fact that CE-LSNNF supports **CE** and CO-LSNNF $\leq_s$ CNF (Proposition 21), it holds that CE-LSNNF $\not\leq_s^*$ CO-LSNNF. □

Lemma 1. *[13] Let c be a non-trivial clause and t a non-trivial term that is equivalent to $\neg c$. Then, $\alpha \models c$ iff $\alpha|t$ is unsatisfiable.*

Proof of Proposition 9. We prove by induction on α. We here only verify the case where α is an $\wedge$-node $\beta_1 \wedge \cdots \wedge \beta_n$. The proof for the case where α is an $\vee$-node $\beta_1 \vee \cdots \vee \beta_n$ is similar to the above case except that we take all β_i's into consideration. The proof for the case where α is a Boolean constant $\top$, $\bot$ or a literal l is trivial.

($\Rightarrow$): Suppose that $\alpha \in$ CE-LSNNF. Clearly, $\alpha \in$ CO-LSNNF. It remains to verify that $\alpha|t$ is CO_{ls} for every non-trivial term t. Let c be a non-trivial clause equivalent to $\neg t$. By Definition 4, if $\alpha \not\models c$, then $\alpha|t$ is satisfiable, and hence $\alpha|t \in$ CO-LSNNF. We now assume that $\alpha \models c$. By Definition 9, there is a conjunct β_i of α s.t. $\beta_i \models c$ and β_i is CE_{ls} w.r.t. c. By the induction hypothesis, we have $\beta_i|t$ is CO_{ls}. By Definition 4, it holds that $\alpha|t$ is CO_{ls}.

($\Leftarrow$): Let $\alpha \in \mathcal{L}$. By assumption, α is CO-logically separable and $\alpha|t$ is also CO-logically separable for every non-trivial term t. Let c' a non-trivial clause and t' a non-trivial term equivalent to $\neg c'$. It remains to verify that α is CE_{ls} w.r.t. c'. α is an $\wedge$-node $\beta_1 \wedge \cdots \wedge \beta_n$: If $\alpha|t'$ is satisfiable, then $\alpha \not\models c'$. It follows from Definition 9 that α is CE_{ls} w.r.t. c'. We now assume that $\alpha|t'$ is unsatisfiable. By Definition 4, there is a conjunct β_i of α s.t. $\beta_i|t'$ is unsatisfiable. By the induction hypothesis, we have that β_i is CE_{ls} w.r.t. c'. Hence, α is CE_{ls} w.r.t. c'. □

Lemma 2. *Unless* NP $\subseteq$ P/poly*, there does not exist a class $\mathcal{L}$ of formulas s.t. every* CNF*-formula (resp.* DNF*-formula) has a polysize equivalent $\mathcal{L}$-formula, and $\mathcal{L}$ supports polytime clausal entailment (resp. term implication).*

Proof. The proof of the result for the case CNF and clausal entailment can be easily extended from that of Theorem 2.5.3.1 in [25]. As for the case DNF and term implication, the proof is a dual version of the above case. We prove by contradiction. Suppose that there exists a language $\mathcal{L}$ s.t. $\mathcal{L} \leq_s$ DNF and $\mathcal{L}$ supports **IM**. Let T be the set of all 3-literal terms that can be generated from $\mathbf{Y} = \{y_1, \ldots, y_n\}$. Let $\mathbf{Z} = \{z_t \mid t \in T\}$ where each z_t is a fresh variable. Let $\beta = \bigvee_{t \in T}(t \wedge z_t)$. We get that the DNF-formula β is of length $O(n^3)$. Let α be a 3-DNF formula over $\mathbf{Y}$. Let $\mathbf{Z}_\alpha = \{z_t \mid t$ is a term of $\alpha\}$. Let u_α denote the term $\bigwedge_{l \in \mathbf{Z}_\alpha} l$. We claim that $\neg\alpha$ is satisfiable iff $u_\alpha \not\models \beta$ (*i.e.* $\neg\beta|u_\alpha = \neg\alpha \wedge \bigwedge_{t \in \beta \setminus \alpha}(\neg t \vee \neg z_t)$ is satisfiable).

($\Rightarrow$): Let ω be an $\mathbf{Y}$-model of $\neg\alpha$ and $\omega' = \omega \cup \{\neg z_t \mid t \in T\}$. Then, ω' is an $(\mathbf{Y} \cup \mathbf{Z})$-model of $\neg\beta|u_\alpha$.

($\Leftarrow$): Since $\neg\beta|u_\alpha \models \neg\alpha$, every model of $\neg\beta|u_\alpha$ is also a model of $\neg\alpha$.

By the assumption that $\mathcal{L} \leq_s$ DNF, there is a polysize compilation function f that map any DNF-formula β to an equivalent $\mathcal{L}$-formula $f(\beta)$. We get that $\neg\alpha$ is satisfiable iff $u_\alpha \not\models f(\beta)$. Since $\mathcal{L}$ supports **IM**, deciding if $\neg\alpha$ is satisfiable, which is a 3-SAT problem, can be determined in time polynomial in the number n of $\mathbf{Y}$. The choice of n is arbitrary and 3-SAT problem is NP-complete, this means that NP $\subseteq$ P/poly. □

Theorem 31. *If $\mathcal{L}'$ supports* **CE** *(resp.* **IM***) and $\mathcal{L} \leq_s$ CNF (resp. $\mathcal{L} \leq_s$ DNF), then $\mathcal{L}' \not\leq_s^* \mathcal{L}$.*

Proof. By Lemma 2 and the transitivity of the succinctness relation, we get that $\mathcal{L}' \leq_s \mathcal{L}$ implies NP $\subseteq$ P/poly, leading to the collapse of the polynomial hierarchy PH [54]. Hence, we get that $\mathcal{L}' \not\leq_s^* \mathcal{L}$. □

Proof of Proposition 10. We first consider the non-membership problem of CE-LSNNF. An NNF-formula α is not a CE-LSNNF-formula iff there exists a non-trivial clause c s.t. $\mathrm{CE}(\alpha, c) = 1$ and $\mathcal{CE}(\alpha, c) = 0$. Since the clausal entailment problem of NNF-formula is coNP-complete, the non-membership problem of CE-LSNNF can be solved by a non-deterministic Turing machine M by invoking an NP oracle that decides if $\mathrm{CE}(\alpha, c) = 1$. Hence, the non-membership problem of CE-LSNNF is in Σ_2^P, and the membership problem is in Π_2^P. □

Proof of Theorem 10. The proof is similar to that of Theorem 7. □

Proof of Theorem 11. The proof is similar to that of Theorem 8. □

Proof of Theorem 12. The proof is similar to that of Theorem 9. □

Proof of Proposition 11. (VA-LSNNF $\leq_s$ IM-LSNNF and IM-LSNNF $\not\leq_s^*$ VA-LSNNF): VA-LSNNF and IM-LSNNF are the duals to CO-LSNNF and VA-LSNNF, respectively. This, together with Proposition 8, imply that VA-LSNNF $\leq_s$ IM-LSNNF and IM-LSNNF $\not\leq_s^*$ VA-LSNNF.

(IM-LSNNF $\not\leq_s^*$ CE-LSNNF and CE-LSNNF $\not\leq_s^*$ IM-LSNNF): By Theorem 31 and the fact that IM-LSNNF supports **IM** and CE-LSNNF $\leq_s$ DNF (Proposition 21), it holds that IM-LSNNF $\not\leq_s^*$ CE-LSNNF. Similarly, we get that CE-LSNNF $\not\leq_s^*$ IM-LSNNF. □

Proof of Proposition 12. The proof is similar to that of Proposition 10. We first consider the non-membership problem of IM-LSNNF. An NNF-formula α is not an IM-LSNNF-formula iff there exists a non-trivial term t s.t. $\mathtt{IM}(\alpha, t) = 1$ and $\mathcal{IM}(\alpha, t) = 0$. Since the term implication problem of NNF-formula is coNP-complete, the non-membership problem of IM-LSNNF can be solved by a non-deterministic Turing machine M by invoking an NP oracle that decides if $\mathtt{IM}(\alpha, c) = 1$. Hence, the non-membership problem of IM-LSNNF is in Σ_2^P, and the membership problem is in Π_2^P. □

Proof of Proposition 13. Since CE-LSNNF $\leq_s$ DNF (resp. IM-LSNNF $\leq_s$ CNF) and LNNF supports **IM** (resp. **CE**) (Proposition 21), it follows from Theorem 31 that LNNF $\not\leq_s^*$ CE-LSNNF (resp. LNNF $\not\leq_s^*$ IM-LSNNF). □

Proof of Theorem 13. Suppose α is an NNF-formula.

Since $\mathcal{CT}(\alpha, \mathbf{Y})$ is a depth-first-search algorithm which is to simply traverse every node of α, the procedure $\mathcal{CT}(\alpha, \mathbf{Y})$ takes $O(|\alpha|)$ time.

We prove that $|\mathrm{Mod}_{\mathbf{Y}}(\alpha)| \leq \mathcal{CT}(\alpha, \mathbf{Y})$ holds.

- α is a literal, $\top$ or $\bot$: It is easily verified that $\mathcal{CT}(\alpha, \mathbf{Y}) = \mathtt{CT}(\alpha, \mathbf{Y})$.
- $\alpha = \beta_1 \wedge \cdots \wedge \beta_n$: Let $\mathbf{Y} = \mathrm{Var}(\alpha)$, $\mathbf{Y}_0 = \mathbf{Y} \setminus \mathrm{Var}(\alpha)$ and $\mathbf{Y}_i = \mathrm{Var}(\beta_i)$ for $1 \leq i \leq n$. It is easily verified that $\mathrm{Mod}_{\mathbf{Y}}(\alpha) \subseteq \mathrm{Mod}_{\mathbf{Y}_0}(\top) \times \mathrm{Mod}_{\mathbf{Y}_1}(\beta_1) \times \cdots \times \mathrm{Mod}_{\mathbf{Y}_n}(\beta_n)$. It follows that $\mathtt{CT}(\alpha, \mathbf{Y}) = |\mathrm{Mod}_{\mathbf{Y}}(\alpha)| \leq \prod_{i=0}^{n} |\mathrm{Mod}_{\mathbf{Y}_i}(\beta_i)|$. By the induction hypothesis, $|\mathrm{Mod}_{\mathrm{Var}(\beta_i)}(\beta_i)| \leq \mathcal{CT}(\beta_i, \mathrm{Var}(\beta_i))$. Hence, $\mathtt{CT}(\alpha, \mathbf{X}) \leq \mathcal{CT}(\alpha, \mathbf{Y})$.
- $\alpha = \beta_1 \vee \cdots \vee \beta_n$: Definitions of $\mathbf{Y}, \mathbf{Y}_0, \mathbf{Y}_1, \cdots, \mathbf{Y}_n$ are the same as the above case. Clearly, $\mathtt{CT}(\alpha, \mathbf{Y}) = |\mathrm{Mod}_{\mathbf{Y}}(\alpha)| = |\bigcup_{i=1}^{n} \mathrm{Mod}_{\mathbf{Y}}(\beta_i)| \leq \sum_i |\mathrm{Mod}_{\mathbf{Y}}(\beta_i)|$. By the induction hypothesis, $|\mathrm{Mod}_{\mathbf{Y}}(\beta_i)| \leq \mathcal{CT}(\beta_i, \mathbf{Y})$. Hence, $\mathtt{CT}(\alpha, \mathbf{Y}) \leq \mathcal{CT}(\alpha, \mathbf{Y})$. □

Lemma 3. *Let α be a satisfiable $\wedge$-node $\beta_1 \wedge \cdots \wedge \beta_n$. The following two conditions are equivalent:*

1. *every two distinct formulas β_i and β_j agree on common variables;*
2. *$Mod_{\mathbf{Y}_1}(\beta_1) \times \cdots \times Mod_{\mathbf{Y}_n}(\beta_n)$ does not contain a pseudo-interpretation where $\mathbf{Y}_i = Var(\beta_i)$.*

Proof. (1 ⇒ 2): We prove by contradiction. Suppose that Item 2 does not hold. Since α is satisfiable, every β_i is satisfiable. Then there exist a variable $x \in \mathrm{Var}(\beta_i) \cap \mathrm{Var}(\beta_j)$ and two distinct β_i and β_j s.t. $x \in \mathbf{L}_i$, $\neg x \in \mathbf{L}_j$, $\mathbf{L}_i \in \mathrm{Mod}_{\mathbf{Y}_i}(\beta_i)$ and $\mathbf{L}_j \in \mathrm{Mod}_{\mathbf{Y}_j}(\beta_j)$. Hence, $\beta_i \not\models \neg x$ and $\beta_j \not\models x$. It contradicts Item 1.

(2 ⇒ 1): We prove by contradiction. Suppose that Item 1 does not hold. Since α is satisfiable, every β_i is satisfiable. There exists $x \in \mathrm{Var}(\beta_i) \cap \mathrm{Var}(\beta_j)$ s.t. $i \neq j$, $\beta_i \not\models x$ and $\beta_j \not\models \neg x$. It follows that $\neg x \in \mathbf{L}_i \in \mathrm{Mod}_{\mathbf{Y}_i}(\beta_i)$ and $x \in \mathbf{L}_j \in \mathrm{Mod}_{\mathbf{Y}_j}(\beta_j)$. So $\{x, \neg x\} \subseteq \mathbf{L} \in \mathrm{Mod}_{\mathbf{Y}_i}(\beta_i) \times \mathrm{Mod}_{\mathbf{Y}_j}(\beta_j)$. It contradicts Item 2. □

Lemma 4. *Let α be an $\vee$-node $\beta_1 \vee \cdots \vee \beta_n$. The following two conditions are equivalent:*

1. *the conjunction of every two distinct disjuncts β_i and β_j is unsatisfiable.*
2. *$Mod_{\mathbf{Y}}(\beta_i) \cap Mod_{\mathbf{Y}}(\beta_j) = \emptyset$ for $i \neq j$ where $\mathbf{Y} = Var(\alpha)$.*

Proof of Theorem 14. (⇒) Let α be a CT-LSNNF-formula and $\mathbf{Y} \supseteq \mathrm{Var}(\alpha)$. We prove that $\mathcal{CT}(\alpha, \mathbf{Y}) = |\mathrm{Mod}_{\mathbf{Y}}(\alpha)| = \mathtt{CT}(\alpha, \mathbf{Y})$.

- α is a literal, $\top$ or $\bot$: It is easily verified that $\mathcal{CT}(\alpha, \mathbf{Y}) = |\mathrm{Mod}_{\mathbf{Y}}(\alpha)|$.
- α is an $\wedge$-node $\beta_1 \wedge \cdots \wedge \beta_n$: Let $\beta_0 = \top$, $\mathbf{Y}_0 = \mathbf{Y} \setminus \mathrm{Var}(\alpha)$ and $\mathbf{Y}_i = \mathrm{Var}(\beta_i)$ for $1 \leq i \leq n$. Obviously, $|\mathrm{Mod}_{\mathbf{Y}_0}(\beta_0)| = 2^{|\mathbf{Y}_0|}$. Suppose that α is satisfiable. By Lemma 3, $\mathrm{Mod}_{\mathbf{Y}_1}(\beta_1) \times \cdots \times \mathrm{Mod}_{\mathbf{Y}_n}(\beta_n)$ does not contain a pseudo-interpretation. Additionally, $\mathrm{Mod}_{\mathbf{Y}_0}(\beta_0) \times \mathrm{Mod}_{\mathbf{Y}_1}(\beta_1) \times \cdots \times \mathrm{Mod}_{\mathbf{Y}_n}(\beta_n)$ does not contain a pseudo-interpretation. Hence, the union of each $\mathbf{Y}_i$-model of β_i is a $\mathbf{Y}$-model of α. By the induction hypothesis, we get that $\mathcal{CT}(\beta_i, \mathbf{Y}_i) = |\mathrm{Mod}_{\mathbf{Y}_i}(\beta_i)|$ for $1 \leq i \leq n$. So $|\mathrm{Mod}_{\mathbf{Y}}(\alpha)| = \prod_{i=0}^{n} |\mathrm{Mod}_{\mathbf{Y}_i}(\beta_i)| = 2^{|\mathbf{Y}_0|} \cdot \prod_{i=1}^{n} \mathcal{CT}(\beta_i, \mathbf{Y}_i) = \mathcal{CT}(\alpha, \mathbf{Y})$. In the case where α is unsatisfiable, then $|\mathrm{Mod}_{\mathbf{Y}}(\alpha)| = 0$. By the assumption that $\alpha \in$ CT-LSNNF, some β_i is unsatisfiable and $\mathtt{CT}_{ls}$. By the induction hypothesis, $\mathcal{CT}(\beta_i, \mathbf{Y}_i) = |\mathrm{Mod}_{\mathbf{Y}_i}(\beta_i)| = 0$. By Definition 14, it follows that $\mathcal{CT}(\alpha, \mathbf{Y}) = 2^{|\mathbf{Y}_0|} \cdot \prod_{i=1}^{n} \mathcal{CT}(\beta_i, \mathbf{Y}_i) = 0 = |\mathrm{Mod}_{\mathbf{Y}}(\alpha)|$.
- α is an $\vee$-node $\beta_1 \vee \cdots \vee \beta_n$: By the induction hypothesis, $\mathcal{CT}(\beta_i, \mathbf{Y}) = |\mathrm{Mod}_{\mathbf{Y}}(\beta_i)|$ for $1 \leq i \leq n$. By the procedure $\mathcal{CT}$, $\mathcal{CT}(\alpha, \mathbf{Y}) = \sum_{i=1}^{n} |\mathrm{Mod}_{\mathbf{Y}}(\beta_i)|$. By Definition 16 and Lemma 4, then $\mathrm{Mod}_{\mathbf{Y}}(\beta_i) \cap \mathrm{Mod}_{\mathbf{Y}}(\beta_j) = \emptyset$. Therefore, $\mathcal{CT}(\alpha, \mathbf{Y}) = \sum_{i=1}^{n} |\mathrm{Mod}_{\mathbf{Y}}(\beta_i)| = |\bigcup_{i=1}^{n} \mathrm{Mod}_{\mathbf{Y}}(\beta_i)| = \mathtt{CT}(\alpha, \mathbf{Y})$.

(⇐) Suppose that the language $\mathcal{L}$ is such that the procedure $\mathcal{CT}$ is CT-overapproximate and CT-underapproximate, that is, for every $\mathcal{L}$-formula α and every set $\mathbf{Y}$ of variables $\mathrm{Var}(\alpha) \subseteq \mathbf{Y}$, we have $\mathtt{CT}(\alpha, \mathbf{Y}) = \mathcal{CT}(\alpha, \mathbf{Y})$. We will prove that every $\mathcal{L}$-formula is CT-logically separable by induction on α.

- α is a literal, $\top$ or $\bot$: It directly follows from Definition 16.
- α is an $\wedge$-node $\beta_1 \wedge \cdots \wedge \beta_n$: We w.l.o.g. assume that $\mathbf{Y} = \mathrm{Var}(\alpha)$. Let $\mathbf{Y}_i = \mathrm{Var}(\beta_i)$ for $1 \leq i \leq n$.
  In the case α is satisfiable, we get that $\mathtt{CT}(\alpha, \mathbf{Y}) > 0$ and $\mathtt{CT}(\beta_i, \mathbf{Y}_i) > 0$ for every $1 \leq i \leq n$. By the assumption $\mathtt{CT}(\alpha, \mathbf{Y}) = \mathcal{CT}(\alpha, \mathbf{Y})$, we prove that:
  1. $\mathtt{CT}(\beta_i, \mathbf{Y}_i) = \mathcal{CT}(\beta_i, \mathbf{Y}_i)$ for every $1 \leq i \leq n$;
  2. $\mathrm{Mod}_{\mathbf{Y}_1}(\beta_1) \times \cdots \times \mathrm{Mod}_{\mathbf{Y}_n}(\beta_n)$ does not contain a pseudo-interpretation.

  We first prove Item 1 by contradiction. Suppose there is a β_j s.t. $\mathcal{CT}(\beta_j, \mathbf{Y}_j) \neq \mathtt{CT}(\beta_j, \mathbf{Y}_j)$. Since the $\mathcal{CT}$ procedure is CT-overapproximate for any NNF-formula (Theorem 13), $\mathcal{CT}(\beta_j, \mathbf{Y}_j) > \mathtt{CT}(\beta_j, \mathbf{Y}_j)$. We get that $\mathcal{CT}(\alpha, \mathbf{Y}) = \prod_{i=1}^{n} \mathcal{CT}(\beta_i, \mathbf{Y}_i) > \prod_{i=1}^{n} |\mathrm{Mod}_{\mathbf{Y}_i}(\beta_i)|$. Since $\mathrm{Mod}_{\mathbf{Y}}(\alpha) \subseteq \mathrm{Mod}_{\mathbf{Y}_1}(\beta_1) \times \cdots \times \mathrm{Mod}_{\mathbf{Y}_n}(\beta_n)$, it holds that $\prod_{i=1}^{n} |\mathrm{Mod}_{\mathbf{Y}_i}(\beta_i)| \geq |\mathrm{Mod}_{\mathbf{Y}}(\alpha)| = \mathtt{CT}(\alpha, \mathbf{Y})$. Hence, we get that $\mathcal{CT}(\alpha, \mathbf{Y}) > \mathtt{CT}(\alpha, \mathbf{Y})$, which contradicts with the assumption that $\mathcal{CT}(\alpha, \mathbf{Y}) = \mathtt{CT}(\alpha, \mathbf{Y})$.
  We now prove Item 2 by contradiction. Suppose $\mathrm{Mod}_{\mathbf{Y}_1}(\beta_1) \times \cdots \times \mathrm{Mod}_{\mathbf{Y}_n}(\beta_n)$ contains a pseudo-interpretation. Then $\prod_{i=1}^{n} |\mathrm{Mod}_{\mathbf{Y}_i}(\beta_i)| > |\mathrm{Mod}_{\mathbf{Y}}(\alpha)| = \mathtt{CT}(\alpha, \mathbf{Y})$. By Item 1, we get that $\mathcal{CT}(\alpha, \mathbf{Y}) = \prod_{i=}^{n} \mathcal{CT}(\beta_i, \mathbf{Y}_i) = \prod_{i=1}^{n} |\mathrm{Mod}_{\mathbf{Y}_i}(\beta_i)|$. Therefore, $\mathcal{CT}(\alpha, \mathbf{Y}) > \mathtt{CT}(\alpha, \mathbf{Y})$, which contradicts with the assumption that $\mathcal{CT}(\alpha, \mathbf{Y}) = \mathtt{CT}(\alpha, \mathbf{Y})$.
  By Item 1 and the induction hypothesis, each β_i is $\mathtt{CT}_{ls}$. By Item 2 and Lemma 3, every two distinct formulas β_i and β_j agree on common variables. It follows from CT-logically separability property that $\alpha \in$ CT-LSNNF.
  In the case where α is unsatisfiable. By the assumption $\mathcal{CT}(\alpha, \mathbf{Y}) = \mathtt{CT}(\alpha, \mathbf{Y})$, $\mathcal{CT}(\alpha, \mathbf{Y}) = \mathtt{CT}(\alpha, \mathbf{Y}) = 0$. By Definition 14, $\mathcal{CT}(\beta_i, \mathbf{Y}_i) = 0$ for some i. Since the $\mathcal{CT}$ procedure is CT-overapproximate (Theorem 13), β_i is unsatisfiable. So $\mathcal{CT}(\beta_i, \mathbf{Y}_i) = \mathtt{CT}(\beta_i, \mathbf{Y}_i)$. By the induction hypothesis, β_i is $\mathtt{CT}_{ls}$. Hence, $\alpha \in$ CT-LSNNF.
- α is an $\vee$-node $\beta_1 \vee \cdots \vee \beta_n$: By the assumption $\mathtt{CT}(\alpha, \mathbf{Y}) = \mathcal{CT}(\alpha, \mathbf{Y})$, we prove that:
  1. $\mathtt{CT}(\beta_i, \mathbf{Y}) = \mathcal{CT}(\beta_i, \mathbf{Y})$ for every $1 \leq i \leq n$;
  2. $\mathrm{Mod}_{\mathbf{Y}}(\beta_i) \cap \mathrm{Mod}_{\mathbf{Y}}(\beta_j) = \emptyset$ for $1 \leq i \neq j \leq n$.

  We first prove Item 1 by contradiction. Suppose there is a β_j s.t. $\mathcal{CT}(\beta_j, \mathbf{Y}) \neq \mathtt{CT}(\beta_j, \mathbf{Y})$. Since the $\mathcal{CT}$ procedure is CT-overapproximate for any NNF-formula (Theorem 13), $\mathcal{CT}(\beta_j, \mathbf{Y}) > \mathtt{CT}(\beta_j, \mathbf{Y})$. We get that $\mathcal{CT}(\alpha, \mathbf{Y}) = \sum_{i=1}^{n} \mathcal{CT}(\beta_i, \mathbf{Y}) > \sum_{i=1}^{n} |\mathrm{Mod}_{\mathbf{Y}}(\beta_i)| \geq \mathtt{CT}(\alpha, \mathbf{Y})$. This contradicts with the assumption that $\mathcal{CT}(\alpha, \mathbf{Y}) = \mathtt{CT}(\alpha, \mathbf{Y})$.
  We now prove Item 2 by contradiction. Suppose there two distinct disjuncts β_j and β_k s.t. $\mathrm{Mod}_{\mathbf{Y}}(\beta_j) \cap \mathrm{Mod}_{\mathbf{Y}}(\beta_k) \neq \emptyset$. Then, $\sum_{i=1}^{n} |\mathrm{Mod}_{\mathbf{Y}}(\beta_i)| > |\mathrm{Mod}_{\mathbf{Y}}(\alpha)| = \mathtt{CT}(\alpha, \mathbf{Y})$. By Item 1, we get that $\mathcal{CT}(\alpha, \mathbf{Y}) = \sum_{i=1}^{n} \mathcal{CT}(\beta_i, \mathbf{Y}) = \sum_{i=1}^{n} |\mathrm{Mod}_{\mathbf{Y}}(\beta_i)|$. Therefore, $\mathcal{CT}(\alpha, \mathbf{Y}) > \mathtt{CT}(\alpha, \mathbf{Y})$, which contradicts with the assumption that $\mathcal{CT}(\alpha, \mathbf{Y}) = \mathtt{CT}(\alpha, \mathbf{Y})$.
  By Item 1 and the induction hypothesis, each β_i is $\mathtt{CT}_{ls}$. By Item 2 and Lemma 4, the conjunction of every two distinct disjuncts β_j and β_k is unsatisfiable. It follows from CT-logically separability property that $\alpha \in$ CT-LSNNF. □

Proof of Proposition 14. (CT-LSNNF $\leq_p$ d-DNNF): Decomposability requires every two distinct conjuncts β_i and β_j for every $\wedge$-node $\alpha = \beta_1 \wedge \cdots \wedge \beta_n$ do not share common variables. Determinism requires the conjunction of every two distinct disjuncts α_i and α_j for every $\vee$-node $\alpha_1 \vee \cdots \vee \alpha_n$ is unsatisfiable. These two constraints corresponds to the requirements for $\wedge$-nodes and $\vee$-nodes of the syntax of CT-LSNNF (cf. Definition 16) respectively.

(d-DNNF $\leq_p$ CT-LSNNF): Suppose that α is in CT-LSNNF.

- α is a literal, $\top$ or $\bot$: Obviously, it satisfies decomposability and determinism.
- $\alpha = \beta_1 \wedge \cdots \wedge \beta_n$: The satisfiability problem of CT-LSNNF-formulas can be solved in polytime. In the case where that α is unsatisfiable, $\bot$ is a d-DNNF-formula equivalent to α and this transformation can be done in polytime. In the case where that α is satisfiable, it follows from the CT-logically separability property that each β_i is in CT-LSNNF. By the induction hypothesis, we let β_i' be an d-DNNF-formula equivalent to β_i. Let $\mathbf{L} = \bigcup_{i=1}^{n} [\mathrm{Lit}(\beta_i') \cap (\bigcup_{j=1}^{i-1} \mathrm{Lit}(\beta_j'))]$ and $\alpha' = \beta_1'|\mathbf{L} \wedge \cdots \wedge \beta_n'|\mathbf{L} \wedge \mathbf{L}$. Since d-DNNF is closed under conditioning (cf. Proposition 5.1 in [9]), $\beta_i'|\mathbf{L}$ is also d-DNNF. It is easily verified that $\alpha' \equiv \alpha$ and α' satisfies determinism and decomposability.
- $\alpha = \beta_1 \vee \cdots \vee \beta_n$: Each β_i is in CT-LSNNF. By the induction hypothesis, we let β_i' be an d-DNNF-formula equivalent to β_i. Let $\alpha' = \beta_1' \vee \cdots \vee \beta_n'$. Since every β_i' satisfies decomposability, α' does. In addition, every β_i' satisfies determinism. It remains to verify that $\beta_i' \wedge \beta_j'$ is unsatisfiable for $i \neq j$. It directly follows from the constraint on $\vee$-node in the syntax of CT-LSNNF. □

Proof of Theorem 15. Let f be the polytime algorithm that transforms any $\mathcal{L}$-formula α into an equivalent CT-LSNNF-formula. By Theorem 14, then $\mathcal{CT}(f(\alpha)) = \mathtt{CT}(\alpha)$. Since f and $\mathcal{CT}$ are executed in polytime, which imply that $\mathcal{L}$ satisfies **CT**. □

Proof of Proposition 15. It is easy to design a deterministic algorithm for the CT-LSNNF membership problem via invoking NP oracle according to the definition of CT-LSNNF-formulas (cf. Definition 16). Identify whether an NNF-formula α is in CT-LSNNF requires checking the satisfiability of some subformulas of α and the conjunction of some subformulas, and checking the entailment of some subformulas and some literals. NP oracles are used to solve the above two checking and are invoked at most $O(n^4)$ times where n is the size of α. □

Proof of Theorem 16. The proof is similar to that of Theorem 13. □

Proof of Theorem 17. The proof is similar to that of Theorem 14. □

Proof of Theorem 18. It is an easy consequence from Theorem 17. □

Proof of Proposition 16. The proof is similar to that of Proposition 15. □

Proof of Theorem 19. We first analyze the time complexity of the procedure $\mathcal{ME}$. Let α be an arbitrary NNF-formula. We prove that the procedure $\mathcal{ME}(\alpha, \mathbf{Y})$ takes $O(|\alpha| \cdot |\mathbf{Y}|^2 \cdot 3^{4\cdot|\mathbf{Y}|})$ time by induction on α.[3]

- α is $\top$, $\bot$ or a literal: It directly follows from the fact that $\mathcal{ME}(\alpha, \mathbf{Y})$ takes at most $O(2^{|\mathbf{Y}|})$ time.[4]
- α is an $\vee$-node $\beta_1 \vee \cdots \vee \beta_n$: By the induction hypothesis, for each β_i, the recursion $\mathcal{ME}(\beta_i, \mathbf{Y})$ takes $O(|\beta_i| \cdot |\mathbf{Y}|^2 \cdot 3^{4\cdot|\mathbf{Y}|})$ time. We now prove that the total union operations over these n sets take $O(n \cdot 3^{2\cdot|\mathbf{Y}|})$ time, which is justified as follows. Firstly, the union operation is performed n times. In addition, each union operation costs $O(3^{2\cdot|\mathbf{Y}|})$ time, since it always act on two sets with size of $O(3^{|\mathbf{Y}|})$. Therefore, the procedure $\mathcal{ME}(\alpha, \mathbf{Y})$ takes $\sum_{1\le i\le n} O(|\beta_i| \cdot |\mathbf{Y}|^2 \cdot 3^{4\cdot|\mathbf{Y}|}) + O(n \cdot 3^{2\cdot|\mathbf{Y}|})$ time. This reduces to $O(\sum_{1\le i\le n} |\beta_i| + n) \cdot O(|\mathbf{Y}|^2 \cdot 3^{4\cdot|\mathbf{Y}|}) = O(|\alpha| \cdot |\mathbf{Y}|^2 \cdot 3^{4\cdot|\mathbf{Y}|})$.
- α is an $\wedge$-node $\beta_1 \wedge \cdots \wedge \beta_n$: By the induction hypothesis, for each β_i, the recursion $\mathcal{ME}(\beta_i, \mathrm{Var}(\beta_i))$ takes $O(|\beta_i| \cdot |\mathbf{Y}|^2 \cdot 3^{4\cdot|\mathbf{Y}|})$ time. Besides, the set $\Omega_{\mathbf{Z}}$ takes $O(2^{|\mathbf{Z}|})$ time, where $\mathbf{Z} = \mathbf{Y} \setminus \mathrm{Var}(\alpha)$. We now prove that the total Cartesian operations over these $n+1$ sets take $O(n \cdot |\mathbf{Y}|^2 \cdot 3^{4\cdot|\mathbf{Y}|})$ time, which is justified as follows. Firstly, the Cartesian product operation is performed $n+1$ times. In addition, each Cartesian operation costs $O(|\mathbf{Y}|^2 \cdot 3^{4\cdot|\mathbf{Y}|})$ time. In details, since it acts on two sets with size of $O(3^{|\mathbf{Y}|})$, we need to perform union operation $O(3^{2\cdot|\mathbf{Y}|})$ times, each of which costs $O(|\mathbf{Y}|^2)$ time. Moreover, an additional operation for eliminating the duplicate elements in the resulting set above, costs $O(3^{4\cdot|\mathbf{Y}|})$ time. Therefore, the procedure $\mathcal{ME}(\alpha, \mathbf{Y})$ takes $\sum_{1\le i\le n} O(|\beta_i| \cdot |\mathbf{Y}|^2 \cdot 3^{4\cdot|\mathbf{Y}|}) + O(n \cdot |\mathbf{Y}|^2 \cdot 3^{4\cdot|\mathbf{Y}|})$ time. This reduces to $O(|\alpha| \cdot |\mathbf{Y}|^2 \cdot 3^{4\cdot|\mathbf{Y}|})$.

We now prove that for any NNF-formula α, $\mathtt{ME}(\alpha, \mathbf{Y}) \subseteq \mathcal{ME}(\alpha, \mathbf{Y})$ by induction on α.

- It directly follows from the procedure $\mathcal{ME}$ (Definition 20) that $\mathtt{ME}(\bot, \mathbf{Y}) = \mathcal{ME}(\bot, \mathbf{Y}) = \{\}$, $\mathtt{ME}(\top, \mathbf{Y}) = \mathcal{ME}(\top, \mathbf{Y}) = \Omega_{\mathbf{Y}}$, $\mathtt{ME}(x, \mathbf{Y}) = \mathcal{ME}(x, \mathbf{Y}) = \{\{x\}\} \times \Omega_{\mathbf{Z}}$ and $\mathtt{ME}(\neg x, \mathbf{Y}) = \mathcal{ME}(\neg x, \mathbf{Y}) = \{\{\neg x\}\} \times \Omega_{\mathbf{Z}}$ where $\mathbf{Z} = \mathbf{Y} \setminus \{x\}$. Therefore, it holds that $\mathtt{ME}(\alpha, \mathbf{Y}) \subseteq \mathcal{ME}(\alpha, \mathbf{Y})$ when α is $\top$, $\bot$ or a literal l.
- α is an $\wedge$-node $\beta_1 \wedge \cdots \wedge \beta_n$: It holds that $\mathrm{Mod}_{\mathbf{Y}}(\alpha) \subseteq \mathrm{Mod}_{\mathbf{Y}_1}(\beta_1) \times \cdots \times \mathrm{Mod}_{\mathbf{Y}_n}(\beta_n) \times \Omega_{\mathbf{Z}}$. By the induction hypothesis, we have that $\mathtt{ME}(\beta_i, \mathbf{Y}_i) = \mathrm{Mod}_{\mathbf{Y}_i}(\beta_i) \subseteq \mathcal{ME}(\beta_i, \mathbf{Y}_i)$ for each β_i. Therefore, it follows that $\mathtt{ME}(\alpha, \mathbf{Y}) \subseteq \mathcal{ME}(\beta_1, \mathbf{Y}_1) \times \cdots \times \mathcal{ME}(\beta_n, \mathbf{Y}_n) \times \Omega_{\mathbf{Z}}$. By the procedure $\mathcal{ME}$, we get that $\mathtt{ME}(\alpha, \mathbf{Y}) \subseteq \mathcal{ME}(\alpha, \mathbf{Y})$.
- α is an $\vee$-node $\beta_1 \vee \cdots \vee \beta_n$: It holds that $\mathrm{Mod}_{\mathbf{Y}}(\alpha) = \bigcup_{i=1}^{n} \mathrm{Mod}_{\mathbf{Y}}(\beta_i)$. By the induction hypothesis, we have that $\mathtt{ME}(\beta_i, \mathbf{Y}) = \mathrm{Mod}_{\mathbf{Y}}(\beta_i) \subseteq \mathcal{ME}(\beta_i, \mathbf{Y})$ for each β_i. Therefore, it follows that $\mathtt{ME}(\alpha, \mathbf{Y}) \subseteq \bigcup_{i=1}^{n} \mathcal{ME}(\beta_i, \mathbf{Y})$. By the procedure $\mathcal{ME}$, we get that $\mathtt{ME}(\alpha, \mathbf{Y}) \subseteq \mathcal{ME}(\alpha, \mathbf{Y})$. □

Proof of Theorem 20. ($\Rightarrow$): Let α be a ME-LSNNF-formula. We w.l.o.g. assume that $\mathbf{Y} = \mathrm{Var}(\alpha)$. We prove that $\mathcal{ME}(\alpha, \mathbf{Y}) = \mathrm{Mod}_{\mathbf{Y}}(\alpha) = \mathtt{ME}(\alpha, \mathbf{Y})$.

- α is $\top$, $\bot$, or a literal l: It is easily verified that $\mathcal{ME}(\alpha, \mathbf{Y}) = \mathtt{ME}(\alpha, \mathbf{Y})$.
- α is an $\wedge$-node $\beta_1 \wedge \cdots \wedge \beta_n$: Suppose that α is satisfiable. By ME-logical separability property (Definition 22) and the induction hypothesis, we get that $\mathcal{ME}(\beta_i, \mathbf{Y}_i) = \mathrm{Mod}_{\mathbf{Y}_i}(\beta_i)$ for each β_i. By ME-logical separability property and Lemma 3, $\mathrm{Mod}_{\mathbf{Y}_1}(\beta_1) \times \cdots \times \mathrm{Mod}_{\mathbf{Y}_n}(\beta_n)$ does not contain a pseudo-interpretation. Hence, the union of each $\mathbf{Y}_i$-model of β_i is a $\mathbf{Y}$-model of α. So it follows from the procedure $\mathcal{ME}$ (Definition 20) that $\mathcal{ME}(\alpha, \mathbf{Y}) = \mathrm{Mod}_{\mathbf{Y}}(\alpha)$.
  In the case where α is unsatisfiable, then $\mathrm{Mod}_{\mathbf{Y}}(\alpha) = \emptyset$. By ME-logical separability property, some β_i is unsatisfiable and $\mathtt{ME}_{ls}$. By the induction hypothesis, $\mathcal{ME}(\beta_i, \mathbf{Y}_i) = \mathrm{Mod}_{\mathbf{Y}_i}(\beta_i) = \emptyset$. It follows from the procedure $\mathcal{ME}$ that $\mathcal{ME}(\alpha, \mathbf{Y}) = \mathcal{ME}(\beta_1, \mathbf{Y}_1) \times \cdots \times \mathcal{ME}(\beta_n, \mathbf{Y}_n) \times \Omega = \emptyset = \mathrm{Mod}_{\mathbf{Y}}(\alpha)$.
- α is an $\vee$-node $\beta_1 \vee \cdots \vee \beta_n$: By ME-logical separability property and the induction hypothesis, we get that $\mathcal{ME}(\beta_i, \mathbf{Y}) = \mathrm{Mod}_{\mathbf{Y}}(\beta_i)$ for each β_i. Since $\mathrm{Mod}_{\mathbf{Y}}(\alpha) = \bigcup_{i=1}^{n} \mathrm{Mod}_{\mathbf{Y}}(\beta_i)$, it follows from the procedure $\mathcal{ME}$ that $\mathcal{ME}(\alpha, \mathbf{Y}) = \mathrm{Mod}_{\mathbf{Y}}(\alpha)$.

($\Leftarrow$): Suppose that the language $\mathcal{L}$ is such that the procedure $\mathcal{ME}$ is ME-overapproximate and ME-underapproximate, that is, for every $\mathcal{L}$-formula α and every set $\mathbf{Y}$ of variables $\mathrm{Var}(\alpha) \subseteq \mathbf{Y}$, we have $\mathtt{ME}(\alpha, \mathbf{Y}) = \mathcal{ME}(\alpha, \mathbf{Y})$. We will prove that every $\mathcal{L}$-formula is ME-logically separable by induction on α.

- α is $\top$, $\bot$ or a literal l: It directly follows from ME-logical separability property (Definition 22).
- α is an $\wedge$-node $\beta_1 \wedge \cdots \wedge \beta_n$: In the case α is satisfiable, we get that $\mathtt{ME}(\alpha, \mathbf{Y}) \neq \emptyset$ and $\mathtt{ME}(\beta_i, \mathbf{Y}_i) \neq \emptyset$ for every β_i. By the assumption $\mathtt{ME}(\alpha, \mathbf{Y}) = \mathcal{ME}(\alpha, \mathbf{Y})$, we prove that:

[3] For simplicity, we assume that the duplicate elements are eliminated in each union and Cartesian product operation over sets. In this case, any intermediate result of $\mathtt{ME}(\alpha, \mathbf{Y})$ has the size bounded by $3^{|\mathbf{Y}|}$.

[4] Although α contains no edges but only a single node, we consider its size $|\alpha|$ as 1 here.

1. $\mathtt{ME}(\beta_i, \mathbf{Y}_i) = \mathcal{ME}(\beta_i, \mathbf{Y}_i)$ for every $1 \leq i \leq n$;
2. $\mathrm{Mod}_{\mathbf{Y}_1}(\beta_1) \times \cdots \times \mathrm{Mod}_{\mathbf{Y}_n}(\beta_n)$ does not contain a pseudo-interpretation.

We first prove Item 1 by contradiction. We w.l.o.g. suppose that $\mathtt{ME}(\beta_1, \mathbf{Y}_1) \neq \mathcal{ME}(\beta_1, \mathbf{Y}_1)$. Since the $\mathcal{ME}$ procedure is ME-overapproximate for any NNF-formula (Theorem 19), $\mathcal{ME}(\beta_1, \mathbf{Y}_1) \supset \mathrm{Mod}_{\mathbf{Y}_1}(\beta_1)$. Moreover, according to the procedure $\mathcal{ME}$, it can be verified that $\mathcal{ME}(\beta_1, \mathbf{Y}_1)$ contains a pseudo-interpretation $\mathbf{L}$. Let $\mathbf{L}' \in \{\mathbf{L}\} \times \mathrm{Mod}_{\mathbf{Y}_2}(\beta_2) \times \cdots \times \mathrm{Mod}_{\mathbf{Y}_n}(\beta_n)$. Clearly, $\mathbf{L}' \notin \mathtt{ME}(\alpha, \mathbf{Y})$. Therefore, we get that $\mathcal{ME}(\alpha, \mathbf{Y}) \supset \mathtt{ME}(\alpha, \mathbf{Y})$, which contradicts with the assumption that $\mathtt{ME}(\alpha, \mathbf{Y}) = \mathcal{ME}(\alpha, \mathbf{Y})$.

We now prove Item 2 by contradiction. Suppose $\mathrm{Mod}_{\mathbf{Y}_1}(\beta_1) \times \cdots \times \mathrm{Mod}_{\mathbf{Y}_n}(\beta_n)$ contains a pseudo-interpretation. Then $\mathrm{Mod}_{\mathbf{Y}_1}(\beta_1) \times \cdots \times \mathrm{Mod}_{\mathbf{Y}_n}(\beta_n) \supset \mathrm{Mod}_{\mathbf{Y}}(\alpha)$. By Item 1, we get that $\mathcal{ME}(\alpha, \mathbf{Y}) \supset \mathrm{Mod}_{\mathbf{Y}}(\alpha)$, which contradicts with the assumption that $\mathtt{ME}(\alpha, \mathbf{Y}) = \mathcal{ME}(\alpha, \mathbf{Y})$.

By Item 1 and the induction hypothesis, each β_i is $\mathtt{ME}_{ls}$. By Item 2 and Lemma 3, every two distinct formulas β_i and β_j agree on common variables. It follows from ME-logically separability property that $\alpha \in$ ME-LSNNF.

- α is an $\vee$-node $\beta_1 \vee \cdots \vee \beta_n$: We first prove that $\mathtt{ME}(\beta_i, \mathbf{Y}) = \mathcal{ME}(\beta_i, \mathbf{Y})$ for every β_i by contradiction. Suppose that there is a β_j s.t. $\mathtt{ME}(\beta_j, \mathbf{Y}) \neq \mathcal{ME}(\beta_j, \mathbf{Y})$. Since the $\mathcal{ME}$ procedure is ME-overapproximate for any NNF-formula (Theorem 19), we get that $\mathcal{ME}(\beta_j, \mathbf{Y}) \supset \mathtt{ME}(\beta_j, \mathbf{Y})$. Moreover, according to the procedure $\mathcal{ME}$, it can be verified that $\mathcal{ME}(\beta_j, \mathbf{Y})$ contains a pseudo-interpretation $\mathbf{L}$. Clearly, $\mathbf{L} \notin \mathtt{ME}(\alpha, \mathbf{Y})$. Therefore, we get that $\mathcal{ME}(\alpha, \mathbf{Y}) \supset \mathtt{ME}(\alpha, \mathbf{Y})$, which contradicts with the assumption that $\mathtt{ME}(\alpha, \mathbf{Y}) = \mathcal{ME}(\alpha, \mathbf{Y})$. By the induction hypothesis, each β_i is $\mathtt{ME}_{ls}$. It follows from ME-logically separability property that $\alpha \in$ ME-LSNNF. □

Proof of Proposition 17. The proof is similar to that of d-DNNF $\sim_p$ CT-LSNNF (Proposition 14) except that we take only decomposability into consideration. □

Proof of Theorem 21. By the fact that DNNF $\sim_p$ ME-LSNNF (Proposition 17) and DNNF supports **ME** (Table 5 in [9]), we get that the language ME-LSNNF permits **ME**. Therefore, if a language $\mathcal{L}$ is polynomially translatable into ME-LSNNF, then $\mathcal{L}$ satisfies **ME**. □

Proof of Proposition 18. The proof is similar to that of checking membership in CT-LSNNF. According to the ME-logical separability property (cf. Definition 22), we can design a deterministic algorithm for the ME-LSNNF membership problem with invoking NP oracle at most $O(n^4)$ times, where n is the size of the input NNF-formula. □

Proof of Theorem 22. The proof is similar to that of Theorem 19. □

Proof of Theorem 23. The proof is similar to that of Theorem 20. □

Proof of Theorem 24. Since CME-LSNNF supports **CME** (Proposition 21), every language polynomially translatable to CME-LSNNF supports **CME** as well. □

Proof of Proposition 19. As ME-LSNNF and CME-LSNNF are a pair of dual languages, we get that for every NNF-formula α, $\alpha \in$ ME-LSNNF iff $\bar{\alpha} \in$ CME-LSNNF. Since the upper bound of checking membership in ME-LSNNF is Δ_2^P (Proposition 18), so is CME-LSNNF. □

Proof of Proposition 20. (ME-LSNNF $\not\leq_s$ CME-LSNNF): According to the property of CME-logical separability (Definition 25), it is easily verified that CME-LSNNF subsumes the language PI. This, together with the fact that ME-LSNNF $\not\leq_s$ PI (Proposition 21), imply that ME-LSNNF is not at least as succinct as CME-LSNNF.

(CME-LSNNF $\not\leq_s$ ME-LSNNF): Since ME-LSNNF and CME-LSNNF (resp. PI and IP) are dual languages, by the fact that ME-LSNNF $\not\leq_s$ PI (Proposition 21), it is easily verified that CME-LSNNF $\not\leq_s$ IP. It follows from ME-LSNNF $\leq_s$ IP (Proposition 21) and the transitivity of succinctness relationship that CME-LSNNF $\not\leq_s$ ME-LSNNF. □

Lemma 5. *Let α be an $\wedge$-node $\beta_1 \wedge \cdots \wedge \beta_n$. Then,* $\mathtt{FO}(\alpha, \mathbf{Y}) \models \bigwedge_{i=1}^n \mathtt{FO}(\beta_i, \mathbf{Y})$.

Proof. It directly follows from definition of forgetting that $\beta_i \models \mathtt{FO}(\beta_i, \mathbf{Y})$ for every $1 \leq i \leq n$. Hence, we get that $\bigwedge_{i=1}^n \beta_i \models \bigwedge_{i=1}^n \mathtt{FO}(\beta_i, \mathbf{Y})$, that is, $\alpha \models \bigwedge_{i=1}^n \mathtt{FO}(\beta_i, \mathbf{Y})$. Since $\bigwedge_{i=1}^n \mathtt{FO}(\beta_i, \mathbf{Y})$ is a formula that does not mention any variable of $\mathbf{Y}$, by definition of forgetting, we obtain that $\mathtt{FO}(\alpha, \mathbf{Y}) \models \bigwedge_{i=1}^n \mathtt{FO}(\beta_i, \mathbf{Y})$. □

Proof of Theorem 25. The analysis of the time complexity of the procedure $\mathcal{FO}$ is similar to that of $\mathcal{CE}$ except that it performs $O(|\mathbf{Y}|)$ work to eliminate the variables of $\mathbf{Y}$ on each terminal node. Therefore, the procedure $\mathcal{FO}(\alpha, \mathbf{Y})$ takes $O(|\alpha| \cdot |\mathbf{Y}|)$ time.

Let α be an arbitrary NNF-formula and $\mathbf{Y} \subseteq \mathbf{X}$. We prove that $\mathtt{FO}(\alpha, \mathbf{Y}) \models \mathcal{FO}(\alpha, \mathbf{Y})$ by induction on α.

- α is $\top$, $\bot$ or a literal l: In the case where α is $\top$, $\bot$ or a literal l where $\mathrm{Var}(l) \notin \mathbf{Y}$, we get that $\mathcal{FO}(\alpha, \mathbf{Y}) = \alpha$. It follows from the definition of forgetting that $\mathtt{FO}(\alpha, \mathbf{Y}) \equiv \alpha$ and hence $\mathtt{FO}(\alpha, \mathbf{Y}) \models \mathcal{FO}(\alpha, \mathbf{Y})$. In the case where $\alpha = l$ where $\mathrm{Var}(l) \in \mathbf{Y}$, we get that $\mathcal{FO}(\alpha, \mathbf{Y}) = \top$. We also obtain $\mathtt{FO}(\alpha, \mathbf{Y}) \models \mathcal{FO}(\alpha, \mathbf{Y})$.

- α is an $\wedge$-node $\beta_1 \wedge \cdots \wedge \beta_n$: By the induction hypothesis, $\mathtt{FO}(\beta_i, \mathbf{Y}) \models \mathcal{FO}(\beta_i, \mathbf{Y})$ for every β_i. Hence, $\bigwedge_{i=1}^{n} \mathtt{FO}(\beta_i, \mathbf{Y}) \models \bigwedge_{i=1}^{n} \mathcal{FO}(\beta_i, \mathbf{Y})$. It follows from Lemma 5 that $\mathtt{FO}(\alpha, \mathbf{Y}) \models \bigwedge_{i=1}^{n} \mathcal{FO}(\beta_i, \mathbf{Y})$. By procedure $\mathcal{FO}$ (Definition 26), we get that $\mathcal{FO}(\alpha, \mathbf{Y}) = \bigwedge_{i=1}^{n} \mathcal{FO}(\beta_i, \mathbf{Y})$, and hence $\mathtt{FO}(\alpha, \mathbf{Y}) \models \mathcal{FO}(\alpha, \mathbf{Y})$.
- α is an $\vee$-node $\beta_1 \vee \cdots \vee \beta_n$: The proof is similar to the above case. □

Lemma 6. *Let α be an* NNF*-formula. Then, $\alpha \models \mathcal{FO}(\alpha, \mathbf{Y})$.*

Proof. We prove that $\alpha \models \mathcal{FO}(\alpha, \mathbf{Y})$ by induction on α.

- α is $\top$, $\bot$ or a literal l: In the case where α is $\top$, $\bot$ or a literal l where $\mathrm{Var}(l) \notin \mathbf{Y}$, we get that $\mathcal{FO}(\alpha, \mathbf{Y}) = \alpha$ and hence $\alpha \models \mathcal{FO}(\alpha, \mathbf{Y})$. In the case where $\alpha = l$ where $\mathrm{Var}(l) \in \mathbf{Y}$, we get that $\mathcal{FO}(\alpha, \mathbf{Y}) = \top$ and hence $\alpha \models \mathcal{FO}(\alpha, \mathbf{Y})$.
- α is an $\wedge$-node $\beta_1 \wedge \cdots \wedge \beta_n$: $\mathcal{FO}(\alpha, \mathbf{Y}) = \mathcal{FO}(\beta_1, \mathbf{Y}) \wedge \cdots \wedge \mathcal{FO}(\beta_n, \mathbf{Y})$. By the induction hypothesis, $\beta_i \models \mathcal{FO}(\beta_i, \mathbf{Y})$ for $1 \leq i \leq n$. So $\alpha \models \mathcal{FO}(\alpha, \mathbf{Y})$.
- α is an $\vee$-node $\beta_1 \vee \cdots \vee \beta_n$: $\mathcal{FO}(\alpha, \mathbf{Y}) = \mathcal{FO}(\beta_1, \mathbf{Y}) \vee \cdots \vee \mathcal{FO}(\beta_n, \mathbf{Y})$. By the induction hypothesis, $\beta_i \models \mathcal{FO}(\beta_i, \mathbf{Y})$ for $1 \leq i \leq n$. It follows that $\beta_i \models \mathcal{FO}(\alpha, \mathbf{Y})$ for $1 \leq i \leq n$. Hence, $\alpha \models \mathcal{FO}(\alpha, \mathbf{Y})$. □

Proof of Theorem 26. ($\Rightarrow$): By the definition of forgetting, it reduces to verify that for every formula β that does not mention any variable of $\mathbf{Y}$, we have $\alpha \models \beta$ iff $\mathcal{FO}(\alpha, \mathbf{Y}) \models \beta$. Any formula β can be equivalently transformed into a CNF-formula. In addition, $\alpha \models c_1 \wedge \cdots \wedge c_n$ iff $\alpha \models c_i$ for $1 \leq i \leq n$. It suffices to prove that for every non-trivial clause c that does not mention any variable of $\mathbf{Y}$, we have $\alpha \models c$ iff $\mathcal{FO}(\alpha, \mathbf{Y}) \models c$.

We first prove the only-if direction. Assume that the formula α is $\mathtt{CE}_{ls}$ w.r.t c and $\alpha \models c$. We prove that $\mathcal{FO}(\alpha, \mathbf{Y}) \models c$ by induction by α.

- α is $\top$, $\bot$ or a literal l: If α is $\top$, $\bot$ or a literal l where $\mathrm{Var}(l) \notin \mathbf{Y}$, then $\mathcal{FO}(\alpha, \mathbf{Y}) = \alpha$ and hence $\mathcal{FO}(\alpha, \mathbf{Y}) \models c$. If $\alpha = l$ where $\mathrm{Var}(l) \in \mathbf{Y}$, then c must be $\top$ since c contains no occurrence of variables of $\mathbf{Y}$. So $\mathcal{FO}(\alpha, \mathbf{Y}) \models c$.
- α is an $\wedge$-node $\beta_1 \wedge \cdots \wedge \beta_n$: By CE-logical separability property, there is some conjunct β_i s.t. $\beta_i \models c$ and β_i is $\mathtt{CE}_{ls}$ w.r.t. c. By the induction hypothesis, it holds that $\mathcal{FO}(\beta_i, \mathbf{Y}) \models c$. It follows that $\mathcal{FO}(\alpha, \mathbf{Y}) \models c$.
- α is an $\vee$-node $\beta_1 \vee \cdots \vee \beta_n$: The proof is similar to the above case except that we take all β_i's into consideration.

We now prove the if-direction. By Lemma 6, we get that $\alpha \models \mathcal{FO}(\alpha, \mathbf{Y})$. This, together with the fact that $\mathcal{FO}(\alpha, \mathbf{Y}_1) \models c$, we get that $\alpha \models c$.

($\Leftarrow$): Suppose that the language $\mathcal{L}$ is such that the procedure $\mathcal{FO}$ is FO-underapproximate and FO-overapproximate, that is, for every $\mathcal{L}$-formula α and every $\mathbf{Y} \subseteq \mathbf{X}$, $\mathtt{FO}(\alpha, \mathbf{Y}) \equiv \mathcal{FO}(\alpha, \mathbf{Y})$. Let c be an arbitrary non-trivial clause and $\mathbf{Z} = \mathrm{Var}(\alpha) \setminus \mathrm{Var}(c)$. We get that $\mathtt{FO}(\alpha, \mathbf{Z}) \models c$ iff $\mathcal{FO}(\alpha, \mathbf{Z}) \models c$. It remains to prove that α is $\mathtt{CE}_{ls}$ w.r.t. c. We prove by induction on α.

- By CE-logical separability property (Definition 9), the Boolean constants $\top$ and $\bot$ and all literals are $\mathtt{CE}_{ls}$ w.r.t. c.
- α is an $\wedge$-node $\beta_1 \wedge \cdots \wedge \beta_n$: In case where $\alpha \not\models c$, it directly follows from CE-logical separability property that α is $\mathtt{CE}_{ls}$ w.r.t. c. In case where $\alpha \models c$. According to the definition of forgetting, it holds that $\alpha \models c$ iff $\mathtt{FO}(\alpha, \mathbf{Z}) \models c$. Since $\mathtt{FO}(\alpha, \mathbf{Z}) \models c$ iff $\mathcal{FO}(\alpha, \mathbf{Z}) \models c$, we get that $\mathcal{FO}(\alpha, \mathbf{Z}) \models c$. By the procedure $\mathcal{FO}$ (Definition 26), $\bigwedge_{i=1}^{n} \mathcal{FO}(\beta_i, \mathbf{Z}) \models c$. Let t be a non-trivial term equivalent to $\neg c$. As $\bigwedge_{i=1}^{n} \mathcal{FO}(\beta_i | t, \mathbf{Z})$ is unsatisfiable and contains no variables, it is easily verified that $\mathcal{FO}(\beta_i | t, \mathbf{Z})$ is unsatisfiable for some i, that is, $\mathcal{FO}(\beta_i, \mathbf{Z}) \models c$. This, together with Theorem 25, implies that $\mathtt{FO}(\beta_i, \mathbf{Z}) \models c$. On the one hand, it follows from definition of forgetting that $\beta_i \models c$. On the other hand, since $\mathcal{FO}(\beta_i, \mathbf{Z}) \models c$ iff $\mathtt{FO}(\beta_i, \mathbf{Z}) \models c$, by the induction hypothesis, β_i is $\mathtt{CE}_{ls}$ w.r.t. c. It follows from CE-logical separability property that α is $\mathtt{CE}_{ls}$ w.r.t. c.
- α is an $\vee$-node $\beta_1 \vee \cdots \vee \beta_n$: The proof is similar to the above case except that we take all β_i's into consideration. □

Lemma 7. *Let α be a* CE-LSNNF*-formula and $\mathbf{Y} \subseteq \mathbf{X}$. Then, $\mathcal{FO}(\alpha, \mathbf{Y})$ is an* CE-LSNNF*-formula.*

Proof. Let $\mathbf{Z} \subseteq \mathbf{Y}$ and β a formula s.t. $\mathrm{Var}(\beta) \subseteq \mathbf{Z}$. It is easily verified that if β is $\mathtt{CE}_{ls}$ w.r.t. every non-trivial clause over $\mathbf{Z}$, then β is in CE-LSNNF. It suffices to prove that $\mathcal{FO}(\alpha, \mathbf{Y})$ is $\mathtt{CE}_{ls}$ w.r.t. every non-trivial clause over $\overline{\mathbf{Y}}$. Let c be a non-trivial clause over $\overline{\mathbf{Y}}$. It follows that the formula α is $\mathtt{CE}_{ls}$ w.r.t. c. We hereafter prove that $\mathcal{FO}(\alpha, \mathbf{Y})$ is $\mathtt{CE}_{ls}$ w.r.t. c by induction on α.

- α is $\top$, $\bot$ or a literal l: It follows directly from CE-logically separability property (Definition 9) that $\mathcal{FO}(\alpha, \mathbf{Y})$ is $\mathtt{CE}_{ls}$ w.r.t. c.
- α is an $\wedge$-node $\beta_1 \wedge \cdots \wedge \beta_n$: In the case where $\alpha \not\models c$, we prove that $\mathcal{FO}(\alpha, \mathbf{Y}) \not\models c$ by contradiction. Assume that $\mathcal{FO}(\alpha, \mathbf{Y}) \models c$. By Lemma 6, we get that $\alpha \models c$. This raises a contradiction. By CE-logically separability property, we get that $\mathcal{FO}(\alpha, \mathbf{Y})$ is $\mathtt{CE}_{ls}$ w.r.t. c.
  In the case where $\alpha \models c$. By CE-logically separability property, we get that some $\beta_i \models c$ and β_i is $\mathtt{CE}_{ls}$ w.r.t. c. By the only-if direction of Theorem 26, $\mathcal{FO}(\beta_i, \mathbf{Y}) \models c$. By the induction hypothesis, we have that $\mathcal{FO}(\beta_i, \mathbf{Y})$ is $\mathtt{CE}_{ls}$ w.r.t. c. From CE-logically separability property and the fact that $\mathcal{FO}(\alpha, \mathbf{Y}) = \mathcal{FO}(\beta_1, \mathbf{Y}) \wedge \cdots \wedge \mathcal{FO}(\beta_n, \mathbf{Y})$, we get that $\mathcal{FO}(\alpha, \mathbf{Y})$ is $\mathtt{CE}_{ls}$ w.r.t. c.
- α is an $\vee$-node $\beta_1 \vee \cdots \vee \beta_n$: The proof is analogical to the above case except that we take all β_i's into consideration. □

Proof of Theorem 27. Assume that $\mathcal{L}$ is a language that is polynomially equivalent to CE-LSNNF. Let f_1 (resp. f_2) be the polytime algorithm that transforms any $\mathcal{L}$-formula (resp. CE-LSNNF-formula) into an equivalent CE-LSNNF-formula (resp. $\mathcal{L}$-formula). Since the procedure $\mathcal{FO}$ is FO-underapproximate and FO-overapproximate for CE-LSNNF (Theorem 26), for any $\mathcal{L}$-formula α and every $\mathbf{Y} \subseteq \mathrm{Var}(\alpha)$, $\mathcal{FO}(f_1(\alpha), \mathbf{Y}) \equiv \mathrm{FO}(\alpha, \mathbf{Y})$. Besides, it follows from Lemma 7 that $\mathcal{FO}(f_1(\alpha), \mathbf{Y})$ is in CE-LSNNF, hence $f_2(\mathcal{FO}(f_1(\alpha), \mathbf{Y}))$ is in $\mathcal{L}$. By the fact that the three algorithms $\mathcal{FO}$, f_1 and f_2 are polytimes, we get that $\mathcal{L}$ satisfies **FO**. □

Proof of Theorem 28. The proof is similar to that of Theorem 25. □

Proof of Theorem 29. The proof is similar to that of Theorem 26. □

Proof of Theorem 30. The proof is similar to that of Theorem 27. □

Definition 29. $\mathrm{CNF}_{\mathrm{CO}}$ is the set of all satisfiable CNF-formulas.

Lemma 8. $\mathrm{CNF}_{\mathrm{CO}}$ *does not satisfy* **CE** *unless* NP = P.

Proof. We prove by contradiction. Assume that $\mathrm{CNF}_{\mathrm{CO}}$ satisfies **CE**. Let α be a CNF-formula of the form $\bigwedge_{i=1}^{n} c_i$ where each c_i is a non-trivial clause. Let $\{x_1, \cdots, x_n\}$ be the set of variables where $x_i \notin \mathrm{Var}(\alpha)$. We construct a satisfiable CNF-formula $\alpha' = \bigwedge_{i=1}^{n}(c_i \vee x_i)$. Let $c = \bigvee_{i=1}^{n} x_i$ and t a term equivalent to $\neg c$. It is easy to show that t is satisfiable and $\alpha'|t \equiv \alpha$. Therefore, it holds that α is unsatisfiable iff $\alpha' \vDash c$. By the fact that $\alpha' \in \mathrm{CNF}_{\mathrm{CO}}$ and the assumption, the satisfiability problem of CNF-formulas is reduced to the clausal entailment problem of $\mathrm{CNF}_{\mathrm{CO}}$-formulas, and would be solved in polytime. Since the satisfiability problem of any CNF-formula is NP-complete, this leads to NP = P. □

Lemma 9. *Let $\mathcal{L}$ be any subset of* NNF*, and $\bar{\mathcal{L}}$ be its dual language. Then $\mathcal{L}$ supports* **CO** *(resp.* **CE/SE/EQ/CT/CCT***) iff $\bar{\mathcal{L}}$ supports* **VA** *(resp.* **IM/SE/EQ/CT/CCT***).*

Proof. We here only verify the statement that $\mathcal{L}$ supports **CO** (resp. **CT**) iff $\bar{\mathcal{L}}$ supports **VA** (resp. **CT**). The other situations can be similarly proved.

1. Every $\bar{\mathcal{L}}$-formula α can be transformed into a $\mathcal{L}$-formula $\bar{\alpha}$ in polytime s.t. $\alpha \equiv \neg\bar{\alpha}$ via De Morgan's laws. It holds that α is valid if and only if $\bar{\alpha}$ is unsatisfiable. Since $\mathcal{L}$ supports **CO**, the validity checking for α can be performed in polytime. Therefore, $\bar{\mathcal{L}}$ supports **VA**. The opposite direction can be similarly proved.
2. Assume that $\mathcal{L}$ supports **CT**. Let $\mathcal{CT}'$ be a polytime model counting procedure for $\mathcal{L}$. Similarly, every $\bar{\mathcal{L}}$-formula α can be transformed its complementary formula $\bar{\alpha} \in \mathcal{L}$ in polytime. Define a new procedure $\mathcal{CT}''$ for $\bar{\mathcal{L}}$ s.t. $\mathcal{CT}''(\alpha, \mathbf{Y}) = 2^{|\mathbf{Y}|} - \mathcal{CT}'(\bar{\alpha}, \mathbf{Y})$. It is easily verified that the procedure $\mathcal{CT}''$ produces a exact number of models of α and can be performed in polytime. Therefore, $\bar{\mathcal{L}}$ supports **CT**. The opposite direction can be similarly proved. □

Proof of Proposition 21. Succinctness: We first prove the succinctness relationships between logical separability based languages and other languages considered in [9,19] shown in Table A.2. The proofs of other succinctness results can be found in [9,19,31,24]. The proof of each cell in Table A.2 with number (i) is illustrated in the i-th item. A number of succinctness relationships between different languages can be verified directly or by applying transitivity of the succinctness relation $\leq_s$.

1. We first prove that LNNF is a subset of CE-LSNNF. Assume that α is in LNNF. We prove that α is in CE-LSNNF by induction on α.
   - α is $\top$, $\bot$ or a literal l: It follows from the property of CE-logically separability (Definition 9) that α is in CE-LSNNF.
   - α is an $\wedge$-node $\beta_1 \wedge \cdots \wedge \beta_n$: By Definition of LNNF (Definition 13), it holds that for any non-trivial clause c, if $\alpha \vDash c$, then $\beta_i \vDash c$ and β_i is in LNNF for some i. By the induction hypothesis, we have β_i is CE_{ls}. It follows from the property of CE-logically separability that α is CE_{ls} w.r.t. any non-trivial clause c, hence α is in CE-LSNNF.
   - α is an $\vee$-node $\beta_1 \vee \cdots \vee \beta_n$: By Definition of LNNF, it holds that each β_i is in LNNF, unconditionally. By the induction hypothesis, we have β_i is CE_{ls}. It follows from the property of CE-logically separability that α is CE_{ls} w.r.t. any non-trivial clause c, hence α is in CE-LSNNF.

   Therefore, CE-LSNNF $\leq_s$ LNNF. Similarly, since LNNF is a subset of IM-LSNNF, we get that IM-LSNNF $\leq_s$ LNNF.
   By Proposition 13, we have that LNNF $\not\leq_s^*$ CE-LSNNF and LNNF $\not\leq_s^*$ IM-LSNNF. It follows from Proposition 8 that CO-LSNNF $\leq_s$ CE-LSNNF and CE-LSNNF $\not\leq_s^*$ CO-LSNNF. By the transitivity of $\leq_s$, we have that CO-LSNNF $\leq_s$ LNNF and LNNF $\not\leq_s^*$ CO-LSNNF.
   It follows from Proposition 11 that VA-LSNNF $\leq_s$ IM-LSNNF, IM-LSNNF $\not\leq_s^*$ VA-LSNNF, CE-LSNNF $\not\leq_s^*$ IM-LSNNF and IM-LSNNF $\not\leq_s^*$ CE-LSNNF.
   By the proof of Proposition 20, we get that CME-LSNNF $\not\leq_s$ IP, ME-LSNNF $\not\leq_s$ CME-LSNNF and CME-LSNNF $\not\leq_s$ ME-LSNNF.
2. We first prove that PI is a subset of LNNF. Let α be a PI-formula of the form $\beta_1 \wedge \cdots \wedge \beta_n$. By Definition of PI, we have that (1) each β_i is a non-trivial clause, which is easy to verified to be in LNNF; (2) for any non-trivial clause c, if $\alpha \vDash c$, then $\beta_j \vDash c$ for some j. It follows from the Definition of LNNF (Definition 13) that α is in LNNF. We can prove that IP is also a subset of

Table A.2
The succinctness relationships between logical separability based languages and other related NNF languages.

	CO-LSNNF	CE-LSNNF	LNNF	CT-LSNNF	VA-LSNNF	IM-LSNNF	ME-LSNNF	CME-LSNNF	CCT-LSNNF
NNF	$\leq_s$ (6), $\geq_s$ (6)	$\leq_s$ (6), $\not\geq_s^*$ (6)	$\leq_s$ (6), $\not\geq_s^*$ (6)	$\leq_s$ (6), $\not\geq_s$ (6)	$\leq_s$ (6), $\geq_s$ (6)	$\leq_s$ (6), $\not\geq_s^*$ (6)	$\leq_s$ (6), $\not\geq_s$ (6)	$\leq_s$ (6), $\not\geq_s$ (4)	$\leq_s$ (9), $\not\geq_s$ (9)
CO-LSNNF	$\leq_s, \geq_s$	$\leq_s$ (1), $\not\geq_s^*$ (1)	$\leq_s$ (1), $\not\geq_s^*$ (1)	$\leq_s$ (6), $\not\geq_s$ (6)	$\leq_s$ (6), $\geq_s$ (6)	$\leq_s$ (6), $\not\geq_s^*$ (6)	$\leq_s$ (6), $\not\geq_s$ (6)	$\leq_s$ (6), $\not\geq_s$ (6)	$\leq_s$ (9), $\not\geq_s$ (9)
VA-LSNNF	$\leq_s$ (6), $\geq_s$ (6)	$\leq_s$ (6), $\not\geq_s^*$ (6)	$\leq_s$ (6), $\not\geq_s^*$ (6)	$\leq_s$ (6), $\not\geq_s$ (6)	$\leq_s, \geq_s$	$\leq_s$ (1), $\not\geq_s^*$ (1)	$\leq_s$ (6), $\not\geq_s$ (6)	$\leq_s$ (6), $\not\geq_s$ (6)	$\leq_s$ (9), $\not\geq_s$ (9)
CE-LSNNF	$\not\geq_s^*$ (1), $\geq_s$ (1)	$\leq_s, \geq_s$	$\leq_s$ (1), $\not\geq_s^*$ (1)	$\leq_s$ (6), $\not\geq_s$ (6)	$\not\geq_s^*$ (6), $\geq_s$ (6)	$\not\geq_s^*$ (1), $\not\geq_s^*$ (1)	$\leq_s$ (6), $\not\geq_s$ (6)	$\not\geq_s^*$ (5), $\not\geq_s$ (4)	?, $\not\geq_s$ (9)
IM-LSNNF	$\not\geq_s^*$ (6), $\geq_s$ (6)	$\not\geq_s^*$ (1), $\not\geq_s^*$ (1)	$\leq_s$ (1), $\not\geq_s^*$ (1)	?, $\not\geq_s$ (3)	$\not\geq_s^*$ (1), $\geq_s$ (1)	$\leq_s, \geq_s$	$\not\geq_s^*$ (6), $\not\geq_s$ (6)	$\leq_s$ (6), $\not\geq_s$ (4)	$\leq_s$ (9), $\not\geq_s$ (9)
LNNF	$\not\geq_s^*$ (1), $\geq_s$ (1)	$\not\geq_s^*$ (1), $\geq_s$ (1)	$\leq_s, \geq_s$	?, $\not\geq_s$ (6)	$\not\geq_s^*$ (6), $\geq_s$ (6)	$\not\geq_s^*$ (1), $\geq_s$ (1)	$\not\geq_s^*$ (6), $\not\geq_s$ (6)	$\not\geq_s^*$ (5), $\not\geq_s$ (4)	?, $\not\geq_s$ (9)
ME-LSNNF	$\not\geq_s$ (6), $\geq_s$ (6)	$\not\geq_s$ (6), $\geq_s$ (6)	$\not\geq_s$ (6), $\not\geq_s^*$ (6)	$\leq_s$ (6), $\not\geq_s$ (6)	$\not\geq_s$ (6), $\geq_s$ (6)	$\not\geq_s$ (6), $\not\geq_s^*$ (6)	$\leq_s, \geq_s$	$\not\geq_s$ (1), $\not\geq_s$ (1)	?, $\not\geq_s$ (9)
DNNF	$\not\geq_s$ (6), $\geq_s$ (6)	$\not\geq_s$ (3), $\geq_s$ (4)	$\not\geq_s$ (3), $\not\geq_s^*$ (5)	$\leq_s$ (6), $\not\geq_s$ (6)	$\not\geq_s$ (6), $\geq_s$ (6)	$\not\geq_s$ (3), $\not\geq_s^*$ (5)	$\leq_s$ (6), $\geq_s$ (6)	$\not\geq_s$ (6), $\not\geq_s$ (6)	?, $\not\geq_s$ (9)
CT-LSNNF	$\not\geq_s$ (6), $\geq_s$ (6)	$\not\geq_s$ (6), $\geq_s$ (6)	$\not\geq_s$ (6), ?	$\leq_s, \geq_s$	$\not\geq_s$ (6), $\geq_s$ (6)	$\not\geq_s$ (6), ?	$\not\geq_s$ (6), $\geq_s$ (6)	$\not\geq_s$ (6), ?	?, ?
d-DNNF	$\not\geq_s$ (6), $\geq_s$ (6)	$\not\geq_s$ (3), $\geq_s$ (4)	$\not\geq_s$ (3), ?	$\leq_s$ (6), $\geq_s$ (6)	$\not\geq_s$ (6), $\geq_s$ (6)	$\not\geq_s$ (3), ?	$\not\geq_s$ (6), $\geq_s$ (6)	$\not\geq_s$ (2), ?	?, ?
CME-LSNNF	$\not\geq_s$ (6), $\geq_s$ (6)	$\not\geq_s$ (4), $\not\geq_s^*$ (5)	$\not\geq_s$ (4), $\not\geq_s^*$ (5)	?, $\not\geq_s$ (6)	$\not\geq_s$ (6), $\geq_s$ (6)	$\not\geq_s$ (6), $\geq_s$ (6)	$\not\geq_s$ (1), $\not\geq_s$ (1)	$\leq_s, \geq_s$	$\leq_s$ (9), $\not\geq_s$ (9)
CCT-LSNNF	$\not\geq_s$ (9), $\geq_s$ (9)	$\not\geq_s$ (9), ?	$\not\geq_s$ (9), ?	?, ?	$\not\geq_s$ (9), $\geq_s$ (9)	$\not\geq_s$ (9), $\geq_s$ (9)	$\not\geq_s$ (9), ?	$\not\geq_s$ (9), $\geq_s$ (9)	$\leq_s, \geq_s$
FBDD	$\not\geq_s$ (6), $\geq_s$ (6)	$\not\geq_s$ (3), $\geq_s$ (4)	$\not\geq_s$ (3), ?	$\not\geq_s$ (6), $\geq_s$ (6)	$\not\geq_s$ (6), $\geq_s$ (6)	$\not\geq_s$ (3), ?	$\not\geq_s$ (6), $\geq_s$ (6)	$\not\geq_s$ (2), ?	$\not\geq_s$ (10), $\geq_s$ (10)
OBDD	$\not\geq_s$ (6), $\geq_s$ (6)	$\not\geq_s$ (3), $\geq_s$ (4)	$\not\geq_s$ (3), ?	$\not\geq_s$ (6), $\geq_s$ (6)	$\not\geq_s$ (6), $\geq_s$ (6)	$\not\geq_s$ (3), ?	$\not\geq_s$ (6), $\geq_s$ (6)	$\not\geq_s$ (2), ?	$\not\geq_s$ (10), $\geq_s$ (10)
$\text{OBDD}_<$	$\not\geq_s$ (6), $\geq_s$ (6)	$\not\geq_s$ (3), $\geq_s$ (4)	$\not\geq_s$ (3), ?	$\not\geq_s$ (6), $\geq_s$ (6)	$\not\geq_s$ (6), $\geq_s$ (6)	$\not\geq_s$ (3), ?	$\not\geq_s$ (6), $\geq_s$ (6)	$\not\geq_s$ (2), ?	$\not\geq_s$ (10), $\geq_s$ (10)
DNF	$\not\geq_s$ (6), $\geq_s$ (6)	$\not\geq_s$ (2), $\geq_s$ (4)	$\not\geq_s$ (2), $\not\geq_s^*$ (5)	$\not\geq_s$ (6), $\not\geq_s^*$ (6)	$\not\geq_s$ (6), $\geq_s$ (6)	$\not\geq_s$ (2), $\not\geq_s^*$ (5)	$\not\geq_s$ (6), $\geq_s$ (6)	$\not\geq_s$ (2), $\not\geq_s$ (4)	$\not\geq_s$ (9), $\not\geq_s$ (9)
CNF	$\not\geq_s$ (6), $\geq_s$ (6)	$\not\geq_s$ (2), $\not\geq_s^*$ (5)	$\not\geq_s$ (2), $\not\geq_s^*$ (5)	$\not\geq_s$ (6), $\not\geq_s$ (6)	$\not\geq_s$ (6), $\geq_s$ (6)	$\not\geq_s$ (2), $\geq_s$ (2)	$\not\geq_s$ (6), $\not\geq_s$ (6)	$\not\geq_s$ (6), $\geq_s$ (2)	$\not\geq_s$ (9), $\not\geq_s^*$ (9)
PI	$\not\geq_s$ (6), $\geq_s$ (6)	$\not\geq_s$ (2), $\geq_s$ (2)	$\not\geq_s$ (2), $\geq_s$ (2)	$\not\geq_s$ (6), $\not\geq_s$ (6)	$\not\geq_s$ (6), $\geq_s$ (6)	$\not\geq_s$ (2), $\geq_s$ (2)	$\not\geq_s$ (6), $\not\geq_s$ (6)	$\not\geq_s$ (2), $\geq_s$ (2)	$\not\geq_s$ (9), ?
IP	$\not\geq_s$ (6), $\geq_s$ (6)	$\not\geq_s$ (2), $\geq_s$ (2)	$\not\geq_s$ (2), $\geq_s$ (2)	$\not\geq_s$ (6), ?	$\not\geq_s$ (6), $\geq_s$ (6)	$\not\geq_s$ (2), $\geq_s$ (2)	$\not\geq_s$ (6), $\geq_s$ (6)	$\not\geq_s$ (2), $\not\geq_s$ (1)	$\not\geq_s$ (9), $\not\geq_s$ (9)
MODS	$\not\geq_s$ (6), $\geq_s$ (6)	$\not\geq_s$ (2), $\geq_s$ (2)	$\not\geq_s$ (2), $\geq_s$ (2)	$\not\geq_s$ (6), $\geq_s$ (6)	$\not\geq_s$ (6), $\geq_s$ (6)	$\not\geq_s$ (2), $\geq_s$ (2)	$\not\geq_s$ (6), $\geq_s$ (6)	$\not\geq_s$ (2), $\geq_s$ (2)	$\not\geq_s$ (10), $\geq_s$ (10)
renH[∨]	$\not\geq_s$ (6), $\geq_s$ (6)	$\not\geq_s$ (7), ?	$\not\geq_s$ (7), $\not\geq_s^*$ (7)	$\not\geq_s$ (6), $\not\geq_s^*$ (6)	$\not\geq_s$ (6), $\geq_s$ (6)	$\not\geq_s$ (7), $\not\geq_s^*$ (7)	$\not\geq_s$ (6), $\not\geq_s$ (6)	$\not\geq_s$ (7), $\not\geq_s$ (7)	$\not\geq_s$ (10), $\not\geq_s$ (9)
K/H[∨]	$\not\geq_s$ (6), $\geq_s$ (6)	$\not\geq_s$ (7), ?	$\not\geq_s$ (7), $\not\geq_s^*$ (7)	$\not\geq_s$ (6), $\not\geq_s^*$ (6)	$\not\geq_s$ (6), $\geq_s$ (6)	$\not\geq_s$ (7), $\not\geq_s^*$ (7)	$\not\geq_s$ (6), $\not\geq_s$ (6)	$\not\geq_s$ (7), $\not\geq_s$ (7)	$\not\geq_s$ (10), $\not\geq_s$ (9)
KROM[∨]	$\not\geq_s$ (6), $\geq_s$ (6)	$\not\geq_s$ (7), $\geq_s$ (8)	$\not\geq_s$ (7), $\not\geq_s^*$ (7)	$\not\geq_s$ (6), $\not\geq_s^*$ (6)	$\not\geq_s$ (6), $\geq_s$ (6)	$\not\geq_s$ (7), $\not\geq_s^*$ (7)	$\not\geq_s$ (6), $\not\geq_s$ (6)	$\not\geq_s$ (7), $\not\geq_s$ (7)	$\not\geq_s$ (10), $\not\geq_s$ (9)
HORN[∨]	$\not\geq_s$ (6), $\geq_s$ (6)	$\not\geq_s$ (7), ?	$\not\geq_s$ (7), $\not\geq_s^*$ (7)	$\not\geq_s$ (6), $\not\geq_s^*$ (6)	$\not\geq_s$ (6), $\geq_s$ (6)	$\not\geq_s$ (7), $\not\geq_s^*$ (7)	$\not\geq_s$ (6), $\not\geq_s$ (6)	$\not\geq_s$ (7), $\not\geq_s$ (7)	$\not\geq_s$ (10), $\not\geq_s$ (9)

LNNF in a similar way by taking non-trivial term and term implication into consideration. Hence, it is an easy consequence that LNNF $\leq_s$ PI and LNNF $\leq_s$ IP.

By the fact that IP $\leq_s$ MODS [9] and MODS $\not\leq_s$ PI [24], we have LNNF $\leq_s$ MODS and MODS $\not\leq_s$ LNNF by the transitivity of $\leq_s$.

Since CNF $\not\leq_s$ IP and CNF $\leq_s$ PI [9], none of CNF and PI is at least as succinct as LNNF. Similarly, none of DNF and IP is at least as succinct as LNNF. Since CE-LSNNF $\leq_s$ LNNF, CNF, DNF, PI, IP and MODS are not at least as succinct as CE-LSNNF. In addition, CE-LSNNF is at least as succinct as PI, IP and MODS, by the transitivity of $\leq_s$.

It is easily verified that CNF is a subset of IM-LSNNF, therefore IM-LSNNF $\leq_s$ CNF. According to the succinctness relations between LNNF and CNF, PI, IP, MODS and DNF, by IM-LSNNF $\leq_s$ LNNF and the transitivity of $\leq_s$, we get that IM-LSNNF is strictly more succinct than CNF, PI, IP and MODS, and that DNF $\not\leq_s$ IM-LSNNF.

It is easily verified that CNF is a subset of CME-LSNNF, therefore CME-LSNNF $\leq_s$ CNF. Since CNF $\leq_s$ PI and CNF $\leq_s$ MODS [9], by the transitivity of $\leq_s$, we get that CME-LSNNF is at least as succinct as PI and MODS. Since d-DNNF, FBDD, OBDD, $\text{OBDD}_<$, DNF, IP, PI and MODS are not at least as succinct as CNF [31,9], it follows from the transitivity of $\leq_s$ that none of them is at least as succinct as CME-LSNNF.

3. As DNNF $\not\leq_s$ PI [31] and LNNF $\leq_s$ PI, we have DNNF $\not\leq_s$ LNNF. By the fact that DNNF $\leq_s$ d-DNNF $\leq_s$ FBDD $\leq_s$ OBDD $\leq_s$ $\text{OBDD}_<$ and the transitivity of $\leq_s$, we have d-DNNF $\not\leq_s$ LNNF, FBDD $\not\leq_s$ LNNF, OBDD $\not\leq_s$ LNNF and $\text{OBDD}_<$ $\not\leq_s$ LNNF.
   By the fact that CE-LSNNF $\leq_s$ LNNF, IM-LSNNF $\leq_s$ LNNF and the transitivity of $\leq_s$, it follows that none of DNNF, d-DNNF, FBDD, OBDD and $\text{OBDD}_<$ is at least as succinct as CE-LSNNF or IM-LSNNF.
4. By the fact that CE-LSNNF $\leq_s$ DNNF (cf. Proposition 7) and the transitivity of $\leq_s$, we get that CE-LSNNF is at least as succinct as DNNF, d-DNNF, FBDD, OBDD, $\text{OBDD}_<$ and DNF.
   Since NNF and DNF are at least as succinct as IP [9], by the fact that CME-LSNNF $\not\leq_s$ IP and the transitivity of $\leq_s$, we get that CME-LSNNF is not at least as succinct as NNF and DNF. Similarly, since CE-LSNNF $\leq_s$ IP, IM-LSNNF $\leq_s$ IP and LNNF $\leq_s$ IP, we get that CME-LSNNF is not at least as succinct as CE-LSNNF, IM-LSNNF and LNNF.
5. By Theorem 31 and the fact that both CE-LSNNF and LNNF support **CE** (cf. Proposition 21), we get that CE-LSNNF $\not\leq_s^*$ CNF and LNNF $\not\leq_s^*$ CNF.
   By the fact that CME-LSNNF $\leq_s$ CNF and the transitivity of $\leq_s$, it holds that CE-LSNNF and LNNF are not at least as succinct as CME-LSNNF, unless PH collapses.
   Similarly, by Theorem 31 and the fact that LNNF and IM-LSNNF support **IM**, we have that LNNF $\not\leq_s^*$ DNF and IM-LSNNF $\not\leq_s^*$ DNF. By the fact that DNNF $\leq_s$ DNF and the transitivity of $\leq_s$, it follows that none of LNNF and IM-LSNNF is at least as succinct as DNNF, unless PH collapses.
6. It is well-known that NNF is at least as succinct as any of other languages in Table A.2 since NNF is their superset [9,19]. It follows from Proposition 2 and Proposition 14 that CO-LSNNF $\sim_s$ NNF and CT-LSNNF $\sim_s$ d-DNNF. Therefore, the succinctness results of CO-LSNNF and other languages are the same as those of NNF and other languages, that is, CO-LSNNF shares the same results of both rows and columns as NNF, and we have the similar result for the case CT-LSNNF and d-DNNF. Similarly, by the fact that VA-LSNNF $\sim_s$ NNF $\sim_s$ CO-LSNNF and ME-LSNNF $\sim_s$ DNNF (cf. Proposition 4 and Proposition 17), VA-LSNNF (resp. ME-LSNNF) shares the same results of both rows and columns as CO-LSNNF (resp. DNNF).
   Since ME-LSNNF and CME-LSNNF (resp. CE-LSNNF and IM-LSNNF) are dual languages, by the fact that CE-LSNNF $\leq_s$ ME-LSNNF, it is easily verified that IM-LSNNF $\leq_s$ CME-LSNNF. Similarly, since DNF $\not\leq_s$ ME-LSNNF, by the property of dual languages, we get that CNF $\not\leq_s$ CME-LSNNF.

Table A.3
The supported polytime queries of logical separability based languages.

	CO	VA	CE	IM	EQ	SE	CT	ME	CME	CCT
CO-LSNNF	√(1)	∘(2)	∘(5)	∘(2)	∘(2)	∘(2)	∘(2)	∘(9)	∘(12)	∘(13)
VA-LSNNF	∘(11)	√(11)	∘(11)	∘(11)	∘(11)	∘(11)	∘(11)	∘(12)	∘(9)	∘(13)
CE-LSNNF	√(1)	∘(2)	√(6)	∘(2)	∘(2)	∘(2)	∘(2)	√(10)	∘(12)	∘(13)
IM-LSNNF	∘(11)	√(11)	∘(11)	√(11)	∘(11)	∘(11)	∘(11)	∘(12)	√(10)	∘(13)
CT-LSNNF	√(3)	√(3)	√(3)	√(3)	?(3)	∘(3)	√(3)	√(3)	√(10)	√(13)
CCT-LSNNF	√(11)	√(11)	√(11)	√(11)	?(11)	∘(11)	√(11)	√(10)	√(10)	√(13)
LNNF	√(1)	√(4)	√(6)	√(4)	∘(7)	∘(7)	∘(8)	√(10)	√(10)	∘(13)
ME-LSNNF	√(3)	∘(3)	√(3)	∘(3)	∘(3)	∘(3)	∘(3)	√(3)	∘(12)	∘(13)
CME-LSNNF	∘(11)	√(11)	∘(11)	√(11)	∘(11)	∘(11)	∘(11)	∘(12)	√(10)	∘(13)

7. By Theorem 31, then none of IM-LSNNF and LNNF is at least as succinct as DNF unless PH collapses. By the fact that renH[∨], K/H[∨], HORN[∨] and KROM[∨] are strictly more succinct than DNF [19] and the transitivity of $\leq_s$, we have IM-LSNNF and LNNF are not at least as succinct as renH[∨], K/H[∨], HORN[∨] and KROM[∨] unless PH collapses. Similarly, by the transitivity of $\leq_s$ and the fact that CME-LSNNF $\leq_s$ DNF, we obtain that CME-LSNNF is not at least as succinct as renH[∨], K/H[∨], HORN[∨] and KROM[∨].
None of renH[∨], K/H[∨], KROM[∨] and HORN[∨] is at least as succinct as PI [19]. This, together with the fact that CE-LSNNF $\leq_s$ PI and the transitivity of $\leq_s$, imply none of them is at least as succinct as CE-LSNNF. Similarly, as LNNF $\leq_s$ PI and IM-LSNNF $\leq_s$ PI and CME-LSNNF $\leq_s$ PI, none of renH[∨], K/H[∨], KROM[∨] and HORN[∨] is at least as succinct as LNNF, IM-LSNNF and CME-LSNNF.
8. We prove that KROM[∨] is polynomially translatable into CE-LSNNF. We first prove that KROM is polynomially translatable into CE-LSNNF. Since KROM is polynomially translatable into PI [27] and PI is a subset of CE-LSNNF, then KROM is polynomially translatable into CE-LSNNF. Let β be an arbitrary KROM[∨]-formula. By the property of KROM[∨], we can split β into a set of KROM-formulas $\{\beta_1, \ldots, \beta_m\}$ in polytime s.t. $\beta \equiv \beta_1 \vee \cdots \vee \beta_m$. Let β_i' be a CE-LSNNF-formula equivalent to β_i, $\beta' = \beta_1' \vee \cdots \vee \beta_m'$. It follows from CE-logically separability property that β' is a CE-LSNNF-formula equivalent to β. Therefore, KROM[∨] is polynomially translatable into CE-LSNNF. Hence, CE-LSNNF $\leq_s$ KROM[∨].
9. CT-LSNNF and CCT-LSNNF (resp. ME-LSNNF and CME-LSNNF) are dual languages. This, together with the fact that ME-LSNNF $\leq_s$ CT-LSNNF and CT-LSNNF $\not\leq_s$ ME-LSNNF, imply that CME-LSNNF $\leq_s$ CCT-LSNNF and CCT-LSNNF $\not\leq_s$ CME-LSNNF. By the transitivity of succinctness, we obtain NNF, CO-LSNNF, VA-LSNNF and IM-LSNNF are at least as succinct as CCT-LSNNF. In a similar way, CCT-LSNNF are not at least as succinct as NNF, CO-LSNNF, VA-LSNNF, CE-LSNNF, IM-LSNNF, LNNF, ME-LSNNF, DNNF, renH[∨], K/H[∨], KROM[∨] and HORN[∨].
CNF and DNF are also dual languages. This, together with the fact that DNF $\not\leq_s$ CT-LSNNF (resp. CT-LSNNF $\not\leq_s^*$ DNF, CNF $\not\leq_s$ CT-LSNNF and CT-LSNNF $\not\leq_s$ CNF), imply that CNF $\not\leq_s$ CCT-LSNNF (resp. CCT-LSNNF $\not\leq_s^*$ CNF, DNF $\not\leq_s$ CCT-LSNNF and CCT-LSNNF $\not\leq_s$ DNF). In a similar way, we can also obtain the succinctness results between CCT-LSNNF and another pair of dual languages: IP and PI.
10. We prove that CCT-LSNNF $<_s$ FBDD. Let $\overline{\text{FBDD}}$ be the dual language to FBDD. We first show that FBDD $\sim_s$ $\overline{\text{FBDD}}$ as follows. Since FBDD supports $\neg$**C** [9], for every FBDD-formula α, there exists a polysize FBDD-formula β s.t. $\beta \equiv \neg\alpha$. As its complementary formula $\bar{\beta}$ is in $\overline{\text{FBDD}}$ and equivalent to α, we get that FBDD $\leq_s$ $\overline{\text{FBDD}}$. The opposite case can be verified similarly. Since CT-LSNNF $<_s$ FBDD and FBDD $\sim_s$ $\overline{\text{FBDD}}$, we get that CT-LSNNF $<_s$ $\overline{\text{FBDD}}$. This, together with the fact that CT-LSNNF and CCT-LSNNF are also dual, implies that CCT-LSNNF $<_s$ FBDD. Since FBDD $\leq_s$ OBDD $\leq_s$ OBDD$_<$ $\leq_s$ MODS [9], by the transitivity of $\leq_s$, it holds that CCT-LSNNF is strictly succinct than each of OBDD, OBDD$_<$ and MODS.
By the fact that none of renH[∨], K/H[∨], KROM[∨] and HORN[∨] is at least as succinct as OBDD$_<$ [19] and CCT-LSNNF $\leq_s$ OBDD$_<$, we get that none of them is at least as succinct as CCT-LSNNF.

Queries: We then prove the supported polytime queries of logical separability based languages shown in Table A.3. The proof of each cell in Table A.3 with number (i) is illustrated in the i-th item.

1. It follows immediately from Theorem 2 that CO-LSNNF supports **CO**. Since LNNF and CE-LSNNF are the subsets of CO-LSNNF, all of them support **CO** as well.
2. It follows immediately since DNNF $\subseteq$ CE-LSNNF $\subseteq$ CO-LSNNF and DNNF does not support **VA**, **IM**, **EQ**, **SE** and **CT** unless NP = P [9].
3. Since CT-LSNNF $\sim_p$ d-DNNF (cf. Proposition 14), CT-LSNNF and d-DNNF possess the same set of queries. Similarly, due to the fact that ME-LSNNF $\sim_p$ DNNF (cf. Proposition 17), ME-LSNNF supports the same set of polytime queries properties as DNNF.
4. Since $t \models \alpha$ iff $\alpha' \models c$ where $\alpha' \equiv \neg\alpha$ and $c \equiv \neg t$, it holds that $\neg$**C** and **CE** imply **IM**. By the fact that LNNF supports $\neg$**C** and **CE** (see below), we get that LNNF supports **IM**. Moreover, since **IM** implies **VA** [9], we obtain that LNNF supports **VA**.
5. Since all satisfiable formulas are CO-LSNNF-formulas, CNF$_{\text{CO}}$ $\subset$ CO-LSNNF. By Lemma 8, CNF$_{\text{CO}}$ does not support **CE** unless NP = P. Neither does CO-LSNNF.
6. It follows immediately from Theorem 8 that CE-LSNNF supports **CE**. Since LNNF is the subset of CE-LSNNF, it support **CE** as well.
7. We first prove that if α is a NNF-formula that contains no positive literal, then α is in LNNF.
   - α is $\top$, $\bot$ or a literal l: It follows from Definition of LNNF (Definition 13) that $\top$, $\bot$ and a negative literal $\neg x$ are in LNNF.
   - α is an $\wedge$-node $\beta_1 \wedge \cdots \wedge \beta_n$: Since each β_i also contains no positive literal, by the induction hypothesis, we get that each β_i is in LNNF. Let c be a non-trivial clause and t a non-trivial term equivalent to $\neg c$. We prove by contradiction. Assume that $\alpha \models c$

Table A.4
The supported polytime transformations of logical separability based languages.

$\mathcal{L}$	CD	FO	SFO	EN	SEN	$\wedge$BC	$\wedge$C	$\vee$BC	$\vee$C	$\neg$C
CO-LSNNF	∘	∘	√	∘	?	∘	∘	√	√	∘
VA-LSNNF	∘	∘	?	∘	√	√	√	∘	∘	∘
CE-LSNNF	√	√	√	∘	?	∘	∘	√	√	∘
IM-LSNNF	√	∘	?	√	√	√	√	∘	∘	∘
LNNF	√	?	?	?	?	∘	∘	∘	∘	√
CT-LSNNF	√	∘	∘	∘	∘	∘	∘	∘	∘	?
CCT-LSNNF	√	∘	∘	∘	∘	∘	∘	∘	∘	?
ME-LSNNF	√	√	√	∘	?	∘	∘	√	√	∘
CME-LSNNF	√	∘	?	√	√	√	√	∘	∘	∘

and each $\beta_i \not\models c$. So $\beta_i|t$ is satisfiable and $\mathrm{Lit}(\beta_i|t)$ is a model of $\beta_i|t$. $\mathrm{Lit}(\alpha|t) = \bigcup_{i=1}^{n} \mathrm{Lit}(\beta_i|t)$ is consistent since α contains no positive literals. It is easily verified that $\mathrm{Lit}(\alpha|t)$ is a model of $\alpha|t$. So $\alpha \not\models c$.

- α is an $\vee$-node $\beta_1 \vee \cdots \vee \beta_n$: Since each β_i also contains no positive literal, by the induction hypothesis, we get that each β_i is in LNNF. Let t be a non-trivial term. We prove by contradiction. Assume that $t \vDash \alpha$ and $t \not\models \beta_i$ for every β_i. So $\neg\beta_i|t$ is satisfiable and $\mathrm{Lit}(\neg\beta_i|t)$ is a model of $\neg\beta_i|t$. $\mathrm{Lit}(\neg\alpha|t) = \bigcup_{i=1}^{n} \mathrm{Lit}(\neg\beta_i|t)$ is consistent since α contains no positive literals. It is easily verified that $\mathrm{Lit}(\neg\alpha|t)$ is a model of $\neg\alpha|t$. So $t \not\models \alpha$.

Let α be a CNF-formula over $\{x_1, \cdots, x_n\}$. Let γ be the formula constructed from α by replacing each occurrence of positive literal x_i in α by the negative literal $\neg x_i'$ where x_i' is a fresh variable. Let $\alpha' = \gamma \wedge \bigwedge_{i=1}^{n}(\neg x_i \vee \neg x_i')$ and $\beta = \bigvee_{i=1}^{n}(\neg x_i \wedge \neg x_i')$. It holds that α is unsatisfiable iff $\alpha' \wedge \beta \equiv \alpha'$ [19]. Since neither $\alpha' \wedge \beta$ nor α' contains positive literal, they are in LNNF. Therefore, the equivalence problem of LNNF-formulas is coNP-hard.

If LNNF supports **SE**, then it supports **EQ**, this contradicts that LNNF does not support **SE** unless P = NP.

8. Since PI is the subset of LNNF and PI does not support **CT** unless NP = P [9]. Neither does LNNF.
9. CO-LSNNF does not support **ME** unless NP = P. We prove by contradiction. Assume that CO-LSNNF satisfies **ME**. Let α be a CNF-formula over $\mathbf{Y} = \{x_1, \ldots, x_n\}$. We construct a new formula $\alpha' = \alpha \vee t$, where $t = x_1 \wedge \cdots \wedge x_n$. Let $\omega = \{x_1, \ldots, x_n\}$. It holds that $\mathrm{Mod}_{\mathbf{Y}}(\alpha') = \mathrm{Mod}_{\mathbf{Y}}(\alpha) \cup \{\omega\}$. Clearly, α' is in CO-LSNNF since α' is satisfiable. By the assumption, the models of α' can be enumerated in polytime. We can enumerate the models of α as follows: if ω is a model of α, then $\mathrm{Mod}_{\mathbf{Y}}(\alpha) = \mathrm{Mod}_{\mathbf{Y}}(\alpha')$; otherwise, $\mathrm{Mod}_{\mathbf{Y}}(\alpha) = \mathrm{Mod}_{\mathbf{Y}}(\alpha') \setminus \omega$. The above procedure can be done in polytime. So model enumeration of CNF would be solved in polytime. However, CNF does not support **ME** unless NP = P [9].
   The proof of the statement that VA-LSNNF does not support **CME** unless NP = P is similar to the above case.
10. Since **CO** and **CD** implies **ME** (cf. Lemma A.3 in [9]) and CE-LSNNF (resp. CCT-LSNNF) supports **CO** and **CD** (see below), it supports **ME**. LNNF also supports **ME** as LNNF is the subset of CE-LSNNF. Similarly, **VA** and **CD** implies **CME** (cf. Proposition P7 in [20]). Since all of IM-LSNNF, CT-LSNNF, CCT-LSNNF, CME-LSNNF and LNNF supports **VA** and **CD**, they support **CME**.
11. It directly follows from Lemma 9 and the fact that VA-LSNNF (resp. IM-LSNNF, CCT-LSNNF, CME-LSNNF) is a dual language to CO-LSNNF (resp. CE-LSNNF, CT-LSNNF, CME-LSNNF).
12. It hold that if a language $\mathcal{L}$ does not satisfy **CO**, then it does not support **ME** (cf. Table 20 in [9]). Since VA-LSNNF, IM-LSNNF and CME-LSNNF do not support **CO**, none of them supports **ME**. Similarly, a language $\mathcal{L}$ that does not satisfy **VA** fails to satisfy **CME**. This, together with the fact that CO-LSNNF, CE-LSNNF and ME-LSNNF do not satisfy **VA**, implies that none of them supports **CME**.
13. Since $\mathrm{CT}(\alpha, \mathbf{Y}) = 2^{|\mathbf{Y}|} - \mathrm{CCT}(\alpha, \mathbf{Y})$, it holds that a language $\mathcal{L}$ supports **CT** iff it supports **CCT**.

Transformations: We finally prove the supported transformations of logical separability based languages shown in Table A.4. By the fact that CT-LSNNF $\sim_p$ d-DNNF (cf. Proposition 14) and ME-LSNNF $\sim_p$ DNNF (cf. Proposition 17), they possess the same set of tractable transformations. So we do not provide the proofs of transformations other than EN and SEN for the languages CT-LSNNF and ME-LSNNF, one can refer to [9] for further details.

- **CD**:
  - CO-LSNNF does not satisfy **CD** unless NP = P.
    We prove by contradiction. Assume that CO-LSNNF satisfies **CD**. By the fact that CO-LSNNF satisfies **CO** (cf. Proposition 21) and that **CO** and **CD** implies **CE** (cf. Lemma A.4 in [9]), we get that CO-LSNNF satisfies **CE**, which contradicts with the fact that its subset $\mathrm{CNF}_{\mathrm{CO}}$ does not satisfy **CE** unless NP = P (cf. Lemma 8).
  - VA-LSNNF does not satisfy **CD** unless NP = P.
    Since VA-LSNNF satisfies **VA** but does not satisfy **IM** unless NP = P (Proposition 21), by the fact that **VA** and **CD** implies **IM** (cf. Lemma A.7 in [9]), we get that VA-LSNNF does not satisfy **CD** unless NP = P.
  - CE-LSNNF satisfies **CD**.
    Let α be a CE-LSNNF-formula and t a satisfiable non-trivial term. We prove that $\alpha|t \in$ CE-LSNNF. By Proposition 9, it remains to prove that $\alpha|t \in$ CO-LSNNF and $(\alpha|t)|t' \in$ CO-LSNNF for every non-trivial term t'. Since $\alpha \in$ CE-LSNNF, we get that $\alpha|t \in$ CO-LSNNF by Proposition 9. Let $t'' = t \wedge \exists \mathrm{Var}(t).t'$. It is easily verified that $(\alpha|t)|t' = \alpha|t''$. Proposition 9 and the fact that $\alpha \in$ CE-LSNNF together imply that $\alpha|t'' \in$ CO-LSNNF. So $(\alpha|t)|t' \in$ CO-LSNNF.

- IM-LSNNF, CCT-LSNNF and CME-LSNNF satisfy **CD**.
  Let f be the polytime algorithm that transforms any NNF-formula α into its complementary formula $\tilde{\alpha}$. Since the languages IM-LSNNF and CE-LSNNF are dual, we get that $\alpha \in$ IM-LSNNF iff $f(\alpha) \in$ CE-LSNNF. Let $\mathcal{CD}'$ be a polytime conditioning algorithm for CE-LSNNF. Given an IM-LSNNF-formula α and a satisfiable non-trivial term t, we now design a new conditioning procedure $\mathcal{CD}''$ where $\mathcal{CD}''(\alpha, t) = f(\mathcal{CD}'(f(\alpha), t))$. It is easily verified that $f(\mathcal{CD}'(f(\alpha), t))$ is equivalent to $\alpha|t$ and is in IM-LSNNF. Therefore, IM-LSNNF satisfies **CD**. Similarly, since ME-LSNNF (resp. CT-LSNNF) supports **CD**, we get that its dual language CME-LSNNF (resp. CCT-LSNNF) satisfies **CD** as well.
- LNNF satisfies **CD**.
  Let c' be a non-trivial clause, t' a non-trivial term equivalent to $\neg c'$. Let t'' be a non-trivial term equivalent to $t \wedge \exists \mathrm{Var}(t).t'$, and c'' a non-trivial clause equivalent to $\neg t''$. Assume that $\alpha \in$ LNNF. It remains to verify that for any satisfiable non-trivial term t, $\alpha|t \in$ LNNF. We prove by induction on α.
  * By the definition of LNNF (Definition 13), $\alpha|t$ is in LNNF since $\alpha|t$ is either a Boolean constant or a literal l.
  * α is an $\wedge$-node $\beta_1 \wedge \cdots \wedge \beta_n$: Suppose that $\alpha|t \vDash c'$. By Lemma 1, we have that $\alpha|t'' = (\alpha|t)|t'$ is unsatisfiable, and thus $\alpha \vDash c''$. By the assumption $\alpha \in$ LNNF and the Definition of LNNF (Definition 13), we get that $\beta_i \vDash c''$ for some i, and thus $\beta_i|t'' = (\beta_i|t)|t'$ is unsatisfiable, that is, $\beta_i|t \vDash c'$. From the assumption that $\alpha \in$ LNNF, it holds that each $\beta_i \in$ LNNF. By the induction hypothesis, we have each $\beta_i|t$ is in LNNF. It follows from the Definition of LNNF that $\alpha|t$ is in LNNF.
  * α is an $\vee$-node $\beta_1 \vee \cdots \vee \beta_n$: The proof is analogical to the $\wedge$-node case above except that we consider term implication instead.

- **FO**:
  - CO-LSNNF does not satisfy **FO** unless NP = P.
    We prove by contradiction. Assume that CO-LSNNF satisfies **FO**. Let α be an CNF-formula and $\mathbf{Y} = \mathrm{Var}(\alpha)$. We construct a new formula $\alpha' = \alpha \vee x$, where $x \notin \mathbf{Y}$. Clearly, α' is in CO-LSNNF since α' is satisfiable. It holds that α is satisfiable iff $\exists \mathbf{Y}.\alpha$ is valid iff $\exists \mathbf{Y}.\alpha'$ is valid. We can compute $\exists \mathbf{Y}.\alpha'$ in polytime by the assumption above, and testing the validity of it also can be done in polytime since it contains only one variable. Therefore, the satisfiability problem of any CNF-formula can be solved in polynomial time, which will lead to NP = P, due to the fact that the satisfiability problem of any CNF-formula is NP-complete.
  - None of VA-LSNNF, IM-LSNNF and CME-LSNNF satisfies **FO** unless NP = P.
    Since α is satisfiable iff $\exists \mathrm{Var}(\alpha).\alpha$ is valid, we get that **FO** implies **CO**. By the fact that VA-LSNNF, IM-LSNNF and CME-LSNNF do not support **CO** (Proposition 21), it then follows that none of them supports **FO** unless NP = P.
  - CE-LSNNF satisfies **FO**.
    It directly follows from Theorem 26.
  - CCT-LSNNF does not satisfy **FO** unless NP = P.
    We first show that for any subset $\mathcal{L}$ of NNF, $\mathcal{L}$ supports **FO** iff its dual language $\tilde{\mathcal{L}}$ supports **EN**. Let f be a polytime algorithm that transforms any NNF-formula α into its complementary formula $\tilde{\alpha}$. Since the languages $\mathcal{L}$ and $\tilde{\mathcal{L}}$ are dual, we get that $\alpha \in \mathcal{L}$ iff $f(\alpha) \in \tilde{\mathcal{L}}$. Assume that $\tilde{\mathcal{L}}$ supports **EN**. Let $\mathcal{EN}'$ be a polytime ensuring algorithm for $\tilde{\mathcal{L}}$. Given a $\mathcal{L}$-formula α and a set $\mathbf{Y}$ of variables, we now design a forgetting procedure $\mathcal{FO}'$ where $\mathcal{FO}'(\alpha, \mathbf{Y}) = f(\mathcal{EN}'(f(\alpha), \mathbf{Y}))$. It is easily verified that $f(\mathcal{EN}'(f(\alpha), \mathbf{Y}))$ is equivalent to $\mathrm{FO}(\alpha, \mathbf{Y})$ and is in $\mathcal{L}$. Therefore, $\mathcal{L}$ supports **FO**. Similarly, we can get the opposite case.
    Since CT-LSNNF does not satisfy **EN** unless NP = P (see below), we get that its dual language CCT-LSNNF does not satisfy **FO** unless NP = P.

- **SFO**:
  - CO-LSNNF satisfies **SFO**.
    Let α be a CO-LSNNF-formula. In the case where α is unsatisfiable. $\exists x.\alpha$ is also unsatisfiable. So $\bot$ is the result of forgetting α from x.
    In the case where α is satisfiable. $\exists x.\alpha$ is also satisfiable. In addition, $\exists x.\alpha \equiv \alpha|x \vee \alpha|\neg x$. Since $\alpha|x \vee \alpha|\neg x$ is also in CO-LSNNF, it is the result of forgetting α from x. Clearly, the above procedure can be done in polytime.
  - CE-LSNNF satisfies **SFO**.
    It directly follows from that CE-LSNNF satisfies **FO**.
  - CCT-LSNNF does not satisfy **SFO** unless NP = P.
    It holds that for any subset $\mathcal{L}$ of NNF, $\mathcal{L}$ supports **SFO** iff its dual language $\tilde{\mathcal{L}}$ supports **SEN**. As CT-LSNNF does not satisfy **SEN** unless NP = P (see below), its dual language CCT-LSNNF does not satisfy **SFO** unless NP = P.

- **EN** and **SEN**:
  - IM-LSNNF and CME-LSNNF satisfy **EN** and **SEN**, and VA-LSNNF satisfies **SEN**.
    It holds that a subset $\mathcal{L}$ of NNF supports **FO** (resp. **SFO**) iff its dual language $\tilde{\mathcal{L}}$ supports **EN** (resp. **SEN**).
    Since CE-LSNNF (resp. ME-LSNNF) supports **FO** and **SFO**, we get that its dual language IM-LSNNF (resp. CME-LSNNF) satisfies **EN** and **SEN**. Since CO-LSNNF supports **SFO**, its dual language VA-LSNNF also satisfies **SEN**.
  - None of CO-LSNNF, VA-LSNNF, CE-LSNNF and ME-LSNNF satisfies **EN** unless NP = P.
    Since VA-LSNNF, CO-LSNNF, IM-LSNNF and CME-LSNNF do not satisfy **FO**, we get that none of their dual languages satisfies **EN** unless NP = P.
  - CT-LSNNF does not satisfy **EN** and **SEN** unless NP = P.
    We assume that CT-LSNNF satisfies **SEN** and prove by contradiction. Let α_1 and α_2 be two CT-LSNNF-formulas and $x \notin \mathrm{Var}(\alpha_1) \cup \mathrm{Var}(\alpha_2)$. It is easily verified that the formula $\alpha = (x \wedge \alpha_1) \vee (\neg x \wedge \alpha_2)$ is in CT-LSNNF. We get that $\forall x.\alpha \equiv \alpha_1 \wedge \alpha_2$. Hence, CT-LSNNF

satisfies $\wedge\mathbf{BC}$. This contradicts with the fact that CT-LSNNF does not satisfy $\wedge\mathbf{BC}$ unless NP = P. Since **EN** implies **SEN**, we get that CT-LSNNF does not satisfy **EN** unless NP = P.

- CCT-LSNNF does not satisfy **EN** or **SEN** unless NP = P.
  Since CT-LSNNF does not satisfy **FO** and **SFO**, we get that its dual language CCT-LSNNF does not satisfy **EN** or **SEN** unless NP = P.

- $\wedge\mathbf{BC}$ and $\wedge\mathbf{C}$:
  - None of CO-LSNNF and CE-LSNNF satisfies $\wedge\mathbf{BC}$ and $\wedge\mathbf{C}$ unless NP = P.
    It is known that checking the satisfiability of the conjunction of two $\mathtt{OBDD}_<$-formulas with different variable orderings $<$ is NP-complete (cf. Lemma 8.14 in [55]). Since $\mathtt{OBDD} \subseteq$ CE-LSNNF $\subseteq$ CO-LSNNF and they all satisfy **CO**, none of them can satisfy $\wedge\mathbf{BC}$ unless NP = P, nor $\wedge\mathbf{C}$.
  - VA-LSNNF, IM-LSNNF and CME-LSNNF satisfy $\wedge\mathbf{BC}$ and $\wedge\mathbf{C}$.
    For any subset $\mathcal{L}$ of NNF, $\mathcal{L}$ supports $\wedge\mathbf{BC}$ (resp. $\wedge\mathbf{C}$) iff its dual $\bar{\mathcal{L}}$ supports $\vee\mathbf{BC}$ (resp. $\vee\mathbf{C}$). As CO-LSNNF (resp. CE-LSNNF, ME-LSNNF) satisfies $\vee\mathbf{BC}$ and $\vee\mathbf{C}$, its dual language VA-LSNNF (resp. IM-LSNNF, CME-LSNNF) satisfies $\wedge\mathbf{BC}$ and $\wedge\mathbf{C}$.
  - LNNF does not satisfy $\wedge\mathbf{BC}$ and $\wedge\mathbf{C}$ unless NP = P.
    Since $\alpha \models \beta$ iff $\alpha \wedge \neg\beta$ is unsatisfiable, we get that $\wedge\mathbf{BC}$, $\neg\mathbf{C}$ and **CO** implies **SE**. Since LNNF satisfies both $\neg\mathbf{C}$ (see below) and **CO** but does not satisfy **SE** unless NP = P, LNNF does not satisfy $\wedge\mathbf{BC}$ unless NP = P, nor $\wedge\mathbf{C}$.
  - CCT-LSNNF does not satisfy $\wedge\mathbf{BC}$ and $\wedge\mathbf{C}$ unless NP = P.
    Since CT-LSNNF does not satisfy $\vee\mathbf{BC}$ and $\vee\mathbf{C}$ unless NP = P, its dual language CCT-LSNNF does not satisfy $\wedge\mathbf{BC}$ and $\wedge\mathbf{C}$ unless NP = P.

- $\vee\mathbf{BC}$ and $\vee\mathbf{C}$:
  - CE-LSNNF and CO-LSNNF satisfy $\vee\mathbf{BC}$ and $\vee\mathbf{C}$.
    It is easy to verify from the property of $\mathtt{CO}_{ls}$ (Definition 4) that the disjunction of CO-LSNNF-formula is still in CO-LSNNF, thus we get that CO-LSNNF satisfies $\vee\mathbf{BC}$ and $\vee\mathbf{C}$. Similar results hold for CE-LSNNF.
  - None of VA-LSNNF, IM-LSNNF, CCT-LSNNF and CME-LSNNF satisfies $\vee\mathbf{BC}$ and $\vee\mathbf{C}$ unless NP = P.
    For any subset $\mathcal{L}$ of NNF, $\mathcal{L}$ supports $\wedge\mathbf{BC}$ (resp. $\wedge\mathbf{C}$) iff its dual $\bar{\mathcal{L}}$ supports $\vee\mathbf{BC}$ (resp. $\vee\mathbf{C}$). This, together with the fact that CO-LSNNF (resp. CE-LSNNF, CT-LSNNF, ME-LSNNF) does not satisfy $\wedge\mathbf{BC}$ and $\wedge\mathbf{C}$, implies that its dual language VA-LSNNF (resp. IM-LSNNF, CCT-LSNNF, CME-LSNNF) does not satisfy $\vee\mathbf{BC}$ and $\vee\mathbf{C}$ unless NP = P.
  - LNNF does not satisfy $\vee\mathbf{BC}$ and $\vee\mathbf{C}$ unless NP = P.
    Since $\alpha_1 \wedge \alpha_2 \equiv \neg(\neg\alpha_1 \vee \neg\alpha_2)$, we get that $\neg\mathbf{C}$ and $\vee\mathbf{BC}$ imply $\wedge\mathbf{BC}$. By the fact that LNNF satisfies $\neg\mathbf{C}$ (see below) but does not satisfy $\wedge\mathbf{BC}$ unless NP = P, it does not satisfy $\vee\mathbf{BC}$ unless NP = P, nor $\vee\mathbf{C}$.

- $\neg\mathbf{C}$:
  - None of CE-LSNNF, CO-LSNNF, VA-LSNNF, IM-LSNNF and CME-LSNNF satisfies $\neg\mathbf{C}$ unless NP = P.
    Since $\neg\mathbf{C}$ and $\vee\mathbf{BC}$ implies $\wedge\mathbf{BC}$, and both of CE-LSNNF and CO-LSNNF satisfy $\vee\mathbf{BC}$ (see above) but do not satisfy $\wedge\mathbf{BC}$ unless NP = P (see above), we get that none of them satisfies $\neg\mathbf{C}$ unless NP = P. Similarly, it holds that none of VA-LSNNF, IM-LSNNF and CME-LSNNF satisfies $\neg\mathbf{C}$ unless NP = P.
  - LNNF satisfies $\neg\mathbf{C}$.
    Suppose that α is in LNNF. It remains to verify that α' is also in LNNF, where α' is an NNF-formula equivalent to $\neg\alpha$ obtained by applying De Morgan's laws. We prove by induction on α.
    - α is $\top$, $\bot$ or a literal l: Clearly, α' is $\bot$, $\top$ or a literal $\neg l$, which is in LNNF.
    - α is an $\wedge$-node $\beta_1 \wedge \cdots \wedge \beta_n$: we get that $\alpha' = \beta'_1 \vee \cdots \vee \beta'_n$, where $\beta'_i \equiv \neg\beta_i$ for every i. By Definition of LNNF, we have (1) for any non-trivial clause c, if $\alpha \models c$, then some $\beta_j \models c$; (2)each β_i is in LNNF. It's an easy consequence that for any non-trivial term t, if $t \models \alpha'$, then some $t \models \beta'_j$. By the induction hypothesis, we have each β'_i is in LNNF. Therefore, it holds that the $\vee$-node α' is in LNNF.
    - α is an $\vee$-node $\beta_1 \vee \cdots \vee \beta_n$: The proof is analogical to the above case. □

References

[1] J. Huang, A. Darwiche, On compiling system models for faster and more scalable diagnosis, in: Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI-2005), 2005, pp. 300–306.
[2] R. Mateescu, R. Dechter, R. Marinescu, AND/OR multi-valued decision diagrams (AOMDDs) for graphical models, J. Artif. Intell. Res. 33 (2008) 465–519.
[3] A. Shih, A. Darwiche, A. Choi, Verifying binarized neural networks by angluin-style learning, in: Proceedings of the Twenty-Second International Conference on Theory and Applications of Satisfiability (SAT-2019), vol. 11628, 2019, pp. 354–370.
[4] A. Darwiche, A. Hirth, On the reasons behind decisions, in: Proceedings of the Twenty-Fourth European Conference on Artificial Intelligence (ECAI-2020), 2020, pp. 712–720.
[5] P.Z.D. Martires, A. Dries, L. De Raedt, Exact and approximate weighted model integration with probability density functions using knowledge compilation, in: Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence (AAAI-2019), 2019, pp. 7825–7833.
[6] A. Shih, A. Choi, A. Darwiche, Compiling Bayesian network classifiers into decision graphs, in: Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence (AAAI-2019), 2019, pp. 7966–7974.
[7] H. Palacios, B. Bonet, A. Darwiche, H. Geffner, Pruning conformant plans by counting models on compiled d-DNNF representations, in: Proceedings of the Fifteenth International Conference on Automated Planning and Scheduling (ICAPS-2005), 2005, pp. 141–150.
[8] J. Huang, Combining knowledge compilation and search for conformant probabilistic planning, in: Proceedings of the Sixteenth International Conference on Automated Planning and Scheduling (ICAPS-2006), 2006, pp. 253–262.
[9] A. Darwiche, P. Marquis, A knowledge compilation map, J. Artif. Intell. Res. 17 (2002) 229–264.
[10] R.E. Bryant, Graph-based algorithms for Boolean function manipulation, IEEE Trans. Comput. 35 (1986) 677–691.

[11] A. Darwiche, SDD: a new canonical representation of propositional knowledge bases, in: Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence (IJCAI-2011), 2011, pp. 819–826.
[12] S. Subbarayan, L. Bordeaux, Y. Hamadi, Knowledge compilation properties of tree-of-BDDs, in: Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence (AAAI-2007), 2007, pp. 502–507.
[13] A. Darwiche, Decomposable negation normal form, J. ACM 48 (2001) 608–647.
[14] H.J. Levesque, A completeness result for reasoning with incomplete first-order knowledge bases, in: Proceedings of the Sixth International Conference on Principles of Knowledge Representation and Reasoning (KR-1998), 1998, pp. 14–23.
[15] J. Qiu, W. Li, Z. Xiao, Q. Guan, L. Fang, Z. Lai, Q. Dong, Knowledge compilation meets logical separability, in: Proceedings of the Thirty-Sixth AAAI Conference on Artificial Intelligence (AAAI-2022), 2022, pp. 5851–5860.
[16] A. Darwiche, On the tractable counting of theory models and its application to truth maintenance and belief revision, J. Appl. Non-Class. Log. 11 (2001) 11–34.
[17] M.R. Krom, The decision problem for formulas in prenex conjunctive normal form with binary disjunctions, J. Symb. Log. 35 (1970) 210–216.
[18] A. Horn, On sentences which are true of direct unions of algebras, J. Symb. Log. 16 (1951) 14–21.
[19] H. Fargier, P. Marquis, Disjunctive closures for knowledge compilation, Artif. Intell. 216 (2014) 129–162.
[20] M. Wachter, R. Haenni, Propositional DAGs: a new graph-based language for representing Boolean functions, in: Proceedings of the Tenth International Conference on Principles of Knowledge Representation and Reasoning (KR-2006), 2006, pp. 277–285.
[21] K. Pipatsrisawat, A. Darwiche, New compilation languages based on structured decomposability, in: Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence (AAAI-2008), 2008, pp. 517–522.
[22] H. Fargier, P. Marquis, Knowledge compilation properties of trees-of-BDDs, revisited, in: Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence (IJCAI-2009), 2009, pp. 772–777.
[23] D.L. Berre, P. Marquis, S. Mengel, R. Wallon, Pseudo-Boolean constraints from a knowledge representation perspective, in: Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI-2018), 2018, pp. 1891–1897.
[24] O. Čepek, M. Chromý, Properties of switch-list representations of Boolean functions, J. Artif. Intell. Res. 69 (2020) 501–529.
[25] B. Selman, H. Kautz, Knowledge compilation and theory approximation, J. ACM 43 (1996) 193–224.
[26] M. Davis, H. Putnam, A computing procedure for quantification theory, J. ACM 7 (1960) 201–215.
[27] P. Marquis, Consequence finding algorithms, in: Handbook of Defeasible Reasoning and Uncertainty Management Systems, Springer, 2000, pp. 41–145.
[28] M. Chavira, A. Darwiche, On probabilistic inference by weighted model counting, Artif. Intell. 172 (2008) 772–799.
[29] A.A. Bulatov, The complexity of the counting constraint satisfaction problem, J. ACM 60 (2013) 1–41.
[30] S. Bova, F. Capelli, S. Mengel, F. Slivovsky, A strongly exponential separation of DNNFs from CNF formulas, arXiv:1411.1995, 2014.
[31] S. Bova, F. Capelli, S. Mengel, F. Slivovsky, Knowledge compilation meets communication complexity, in: Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI-2016), 2016, pp. 1008–1014.
[32] J. Lang, P. Liberatore, P. Marquis, Propositional independence: formula-variable independence and forgetting, J. Artif. Intell. Res. 18 (2003) 391–443.
[33] L. Fang, H. Wan, X. Liu, B. Fang, Z. Lai, Dependence in propositional logic: formula-formula dependence and formula forgetting - application to belief update and conservative extension, in: Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-2018), 2018, pp. 1835–1844.
[34] A. Cimatti, M. Roveri, P. Bertoli, Conformant planning via symbolic model checking and heuristic search, Artif. Intell. 159 (2004) 127–206.
[35] D. Bryce, S. Kambhampati, D.E. Smith, Planning graph heuristics for belief space search, J. Artif. Intell. Res. 26 (2006) 35–99.
[36] J. Lagniez, E. Lonca, P. Marquis, Definability for model counting, Artif. Intell. 281 (2020) 103229.
[37] A. Darwiche, P. Marquis, On quantifying literals in Boolean logic and its applications to explainable AI, J. Artif. Intell. Res. 72 (2021) 285–328.
[38] H. Fargier, P. Marquis, Extending the knowledge compilation map: closure principles, in: Proceedings of the Eighth European Conference on Artificial Intelligence (ECAI-2008), 2008, pp. 50–54.
[39] F. Koriche, J.-M. Lagniez, P. Marquis, S. Thomas, Knowledge compilation for model counting: affine decision trees, in: Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence (IJCAI-2013), 2013, pp. 947–953.
[40] A. Bart, F. Koriche, J.-M. Lagniez, P. Marquis, Symmetry-driven decision diagrams for knowledge compilation, in: Proceedings of the Twenty-First European Conference on Artificial Intelligence (ECAI-2014), 2014, pp. 51–56.
[41] G.V. den Broeck, A. Darwiche, On the role of canonicity in knowledge compilation, in: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI-2015), 2015, pp. 1641–1648.
[42] S. Bova, SDDs are exponentially more succinct than OBDDs, in: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-2016), 2016, pp. 929–935.
[43] Y. Lai, K. Meel, R. Yap, The power of literal equivalence in model counting, in: Proceedings of the Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI-2021), 2021, pp. 3851–3859.
[44] P. Shah, A. Bansal, S. Akshay, S. Chakraborty, A normal form characterization for efficient Boolean Skolem function synthesis, in: Proceedings of the Thirty-Sixth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS-2021), 2021, pp. 1–13.
[45] A. Darwiche, New advances in compiling CNF into decomposable negation normal form, in: Proceedings of the Sixteenth European Conference on Artificial Intelligence (ECAI-2004), 2004, pp. 328–332.
[46] U. Oztok, A. Darwiche, An exhaustive DPLL algorithm for model counting, J. Artif. Intell. Res. (2018) 1–32.
[47] C. Muise, S. McIlraith, J. Beck, E. Hsu, DSHARP: fast d-DNNF compilation with sharpSAT, in: Proceedings of the Twenty-Fifth Canadian Conference on Artificial Intelligence, 2012, pp. 356–361.
[48] J.-M. Lagniez, P. Marquis, An improved decision-DNNF compiler, in: Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI-2017), 2017, pp. 667–673.
[49] J. Huang, A. Darwiche, The language of search, J. Artif. Intell. Res. (2007) 91–219.
[50] F. Bacchus, J. Winter, Effective preprocessing with hyper-resolution and equality reduction, in: Proceedings of the Seventh International Conference on Theory and Applications of Satisfiability Testing (SAT-2003), 2003, pp. 341–355.
[51] J.-M. Lagniez, P. Marquis, Preprocessing for propositional model counting, in: Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence (AAAI-2014), 2014, pp. 2688–2694.
[52] S. Akshay, S. Chakraborty, S. Goel, S. Kulal, S. Shah, What's Hard About Boolean Functional Synthesis?, Proceedings of the Thirtieth International Conference on Computer Aided Verification (CAV-2018), vol. 10981, Springer, 2018, pp. 251–269.
[53] S. Akshay, J. Arora, S. Chakraborty, S. Krishna, D. Raghunathan, S. Shah, Knowledge compilation for Boolean functional synthesis, in: Proceedings of 2019 International Conference on Formal Methods in Computer Aided Design (FMCAD-2019), 2019, pp. 161–169.
[54] R.M. Karp, R.J. Lipton, Some connections between nonuniform and uniform complexity classes, in: Proceedings of the Twelfth Annual ACM Symposium on Theory of Computing (STOC-1980), 1980, pp. 302–309.
[55] C. Meinel, T. Theobald, Algorithms and Data Structures in VLSI Design: OBDD Foundations and Applications, Springer, 1998.