Disjunctive closures for knowledge compilation

Hélène Fargier, Pierre Marquis

Disjunctive closures for knowledge compilation ☆

Hélène Fargier [a], Pierre Marquis [b,*]

[a] IRIT, CNRS, Université Paul Sabatier, Toulouse, France
[b] CRIL, CNRS, Université d'Artois, Lens, France

A B S T R A C T

The knowledge compilation (KC) map [1] can be viewed as a multi-criteria evaluation of a number of target classes of representations for propositional KC. Using this map, the choice of a class for a given application can be made, considering both the space efficiency of it (i.e., its ability to represent information using little space), and its time efficiency, i.e., the queries and transformations which can be achieved in polynomial time, among those of interest for the application under consideration. When no class of propositional representations offers all the transformations one would expect, some of them can be left *implicit*. This is the key idea underlying the concept of *closure* introduced here: instead of performing computationally expensive transformations, one just remembers that they have to be done. In this paper, we investigate the disjunctive closure principles, i.e., disjunction, existential quantification, and their combinations. We provide several characterization results concerning the corresponding closures. We also extend the KC map with new propositional languages obtained as disjunctive closures of several incomplete propositional languages, including the well-known KROM (the CNF formulae containing only binary clauses), HORN (the CNF formulae containing only Horn clauses), and AFF (the affine language, which is the set of conjunctions of XOR-clauses). Each introduced language is evaluated along the lines of the KC map.

Keywords:
Knowledge representation
Knowledge compilation
Computational complexity

1. Introduction

Knowledge compilation (KC) consists of a family of approaches which aim at improving the efficiency of some computational tasks – typically, saving on-line computation time – via pre-processing. The pre-processing step consists in turning some pieces of available information into a compiled form, during an off-line compilation phase.

KC gathers a number of research lines focusing on different problems, [5–12], ranging from theoretical ones, where the key question is the compilability issue, i.e., determining whether pre-processing can lower the computational complexity of some tasks, to more practical ones, especially the design of compilation algorithms for some specific tasks like clausal entailment. An important research line [13,1] is concerned with the issue of choosing a target class of representations for KC. In [1], the authors argue that the choice of a target class for a compilation purpose must be based both on its time efficiency, defined as the set of queries and transformations which can be achieved in polynomial time when the pieces of data to be exploited are represented in the class, as well as the space efficiency of the class, i.e., its ability to represent data using little space. Thus, the KC map [1] is an evaluation of a dozen of significant propositional classes $\mathcal{L}$, also called propositional fragments, w.r.t. several dimensions: the space efficiency (aka succinctness) of the fragment and its time

☆ This paper is an extended and revised version of [2,3]. It also elaborates on some of the results presented in [4].

* Corresponding author.
E-mail addresses: fargier@irit.fr (H. Fargier), marquis@cril.univ-artois.fr (P. Marquis).

efficiency (aka tractability), i.e., the class of queries and transformations it supports (or not) in polynomial time, often under standard assumptions from complexity theory. The KC map is intended to serve as a guide for selecting the "right" target class given the requirements imposed by the application under consideration.

The KC map reported in [1] has already been extended to other propositional classes, queries and transformations in a number of subsequent papers, including [14–18,2,4,19–21,3,22]. In all those papers, queries and transformations are also viewed as properties of classes of propositional representations $\mathcal{L}$. One says that $\mathcal{L}$ offers or satisfies a given query or a transformation when there exists a polynomial-time algorithm to achieve it, provided that the input representations are from $\mathcal{L}$. When such an algorithm does not exist for sure or unless $\mathsf{P} = \mathsf{NP}$, one says that $\mathcal{L}$ does not offer the query or the transformation.

When no class of propositional representations offers all the transformations one would expect, an approach consists in leaving some of them *implicit*. This is the key idea underlying *closure principles* as introduced in this paper: instead of performing some computationally expensive transformations on representations, one just remembers in the representations that the transformations have to be done. This leads to extend the previous classes to new ones, which are at least as succinct, and for which implicit transformations are for free. Another nice effect of some implicit transformations on incomplete propositional languages is to recover completeness, i.e., the ability to represent any Boolean function.

In this paper, we investigate the *disjunctive closure principles*, i.e., disjunction [$\vee$], existential quantification [$\exists$], and their combinations. The disjunction principle [$\vee$] when applied to a class $\mathcal{L}$ of representations leads to a class $\mathcal{L}[\vee]$, the disjunction closure of $\mathcal{L}$, which qualifies disjunctions of representations from $\mathcal{L}$, while the existential quantification principle [$\exists$] applied to a class $\mathcal{L}$ leads to a class $\mathcal{L}[\exists]$, the existential closure of $\mathcal{L}$, which qualifies existentially quantified representations from $\mathcal{L}$. $\mathcal{L}[\vee, \exists]$, the full disjunctive closure of $\mathcal{L}$, is obtained by applying both disjunctive closure principles to $\mathcal{L}$. We provide a number of characterization results concerning the corresponding closures. Especially, we show that applying at most once each disjunctive closure principle on $\mathcal{L}$ is enough, in the sense that applying one of them twice or more leads to classes polynomially equivalent to $\mathcal{L}$. We also identify the queries and transformations which are preserved by applying disjunctive closure principles.

In addition, we extend the KC map with new classes of propositional representations obtained as disjunctive closures of several incomplete propositional languages, namely the well-known Krom CNF fragment KROM (also known as the bijunctive fragment) [23] the Horn CNF fragment HORN [24], and the affine fragment AFF (also known as the biconditional fragment) [25], as well as K/H (Krom or Horn CNF formulae) and renH, the language of renamable Horn CNF formulae [26]. Each of these languages is a well-known polynomial class for the satisfiability problem SAT (i.e., it offers **CO**), but none of them is fully expressive w.r.t. propositional logic (there exist propositional formulae which cannot be represented in any of them), which drastically restricts their attractiveness for the KC purpose. Importantly, switching from any of those languages to its disjunction closure or to its full disjunctive closure leads to recover a fully expressive propositional language. This is crucial for many applications.

The rest of the paper is organized as follows. In Section 2, some formal preliminaries about graph-based, quantified, propositional representations are provided. In Section 3, we make precise the queries and transformations of interest, and extend the notions of expressiveness, succinctness and polynomial translations to any subsets of the class of graph-based, quantified, propositional representations. In Section 4, the concepts of disjunctive closures of a class of propositional representations are defined and we derive a number of characterization results about them. In Section 5, the disjunctive closures of KROM, HORN, K/H, renH, and AFF are considered and we analyze them along the lines of the KC map. Finally, Section 6 concludes the paper by discussing the results, pointing out the disjunctive closures which appear as the best target classes for the KC purpose; it also gives some perspectives for further research.

2. Quantified propositional representations

2.1. Syntax

In this paper, we consider subsets of the class C-QDAG of quantified propositional representations over a countably infinite set PS of propositional variables, given a finite set C of propositional connectives. Each connective $c \in C$ is supposed to have a fixed, finite arity. Leaf nodes of such DAGs are labeled by literals, where a literal (over $V \subseteq PS$) is an element $x \in V$ (a positive literal) or a negated one $\neg x$ (a negative literal), or a Boolean constant ($\top$ and $\bot$). L_V is the set of all literals over V. Literal $\dot{l}$ is the complementary literal of literal l, so that $\overline{\top} = \bot$, $\overline{\bot} = \top$, $\bar{x} = \neg x$ and $\overline{\neg x} = x$. For a literal l different from a Boolean constant, $var(l)$ denotes the corresponding variable: for $x \in PS$, we have $var(x) = x$ and $var(\neg x) = x$.

Formally, C-QDAG is given by:

Definition 1 *(C-QDAG).* C-QDAG is the set of all finite, single-rooted DAGs (also referred to as "representations") α where:

- each leaf node of α is labeled by a literal l over PS,
- each internal node of α is labeled by a connective $c \in C$ and has as many children as required by c (it is then called a c-node), or is labeled by a quantification $\exists x$ or $\forall x$ (where $x \in PS$) and has a single child.[1]

[1] Each binary connective c which is associative (like $\wedge$, $\vee$, $\oplus$) corresponds to a family of connectives (with the same name c) of arity i with $i \geq 2$. For each $i \geq 2$, the connective c of arity i is defined by: for every i-tuple $\langle x_1, \ldots, x_i \rangle$ of Boolean values, $c(x_1, \ldots, x_i) = c(x_1, c(x_2, c(\ldots, c(x_{i-1}, x_i)) \ldots))$.

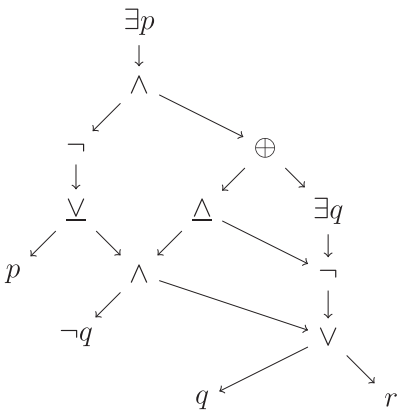

Fig. 1. A C-QDAG representation with $C = \{\wedge, \vee, \neg, \oplus\}$.

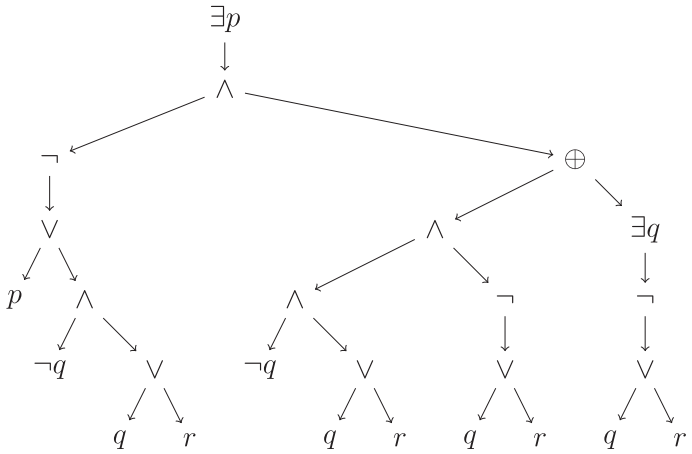

Fig. 2. A C-QDAG formula with $C = \{\wedge, \vee, \neg, \oplus\}$.

The size $|\alpha|$ of a C-QDAG representation α is the number of nodes plus the number of arcs in the DAG. $Var(\alpha)$ denotes the set of *free* variables of α, i.e., those variables x for which there exists a leaf node N_x of α labelled by a literal l such that $var(l) = x$ and there is a path from the root of α to N_x such that no node from it is labelled by $\exists x$ or $\forall x$. Clearly enough, determining whether a given $x \in PS$ belongs to $Var(\alpha)$ can be done in time polynomial in the size of α[2]; similarly, computing $Var(\alpha)$ can also be achieved in time polynomial in the size of α.

Fig. 1 presents a C-QDAG representation α with $C = \{\wedge, \vee, \neg, \oplus\}$. Its set of free variables is $Var(\alpha) = \{q, r\}$.

As Fig. 1 exemplifies it, a C-QDAG mainly corresponds to a Quantified Boolean Circuit [27]. Abusing words, such DAG-based representations are also referred to as "formulae" in the KC literature, and classes of such representations are called "languages". In the following, we will only use the term "formula" for designating a tree-shaped representation of a Boolean function, and the term "language" for sets of formulae. Fig. 2 gives a C-QDAG formula with $C = \{\wedge, \vee, \neg, \oplus\}$.

Many classes of propositional representations considered so far as target classes for KC are subsets of C-QDAG with $C = \{\wedge, \vee, \neg, \oplus\}$, and typically subsets of C-DAG, the subset of C-QDAG with $C = \{\wedge, \vee, \neg, \oplus\}$ where no node labeled by a quantification is allowed. Especially, the propositional DAGs considered in [14] are C-DAG representations with $C = \{\wedge, \vee, \neg\}$, and the classes considered in [1] are subsets of DAG-NNF (the non-quantified DAGs with $C = \{\wedge, \vee\}$). Clearly enough, for each non-quantified representation α from C-DAG, $Var(\alpha)$ coincides with the set of variables occurring in α.

In Fig. 1, the DAG rooted at the $\underline{\wedge}$ node is a C-DAG representation with $C = \{\wedge, \vee, \neg\}$ and the DAG rooted at the $\underline{\vee}$ node is a DAG-NNF representation. DNNF is the subset of DAG-NNF consisting of DAGs where each $\wedge$-node $\wedge(\alpha_1, \ldots, \alpha_k)$ is decomposable, which means that $\forall i, j \in \{1, \ldots, k\}$, if $i \neq j$ then $Var(\alpha_i) \cap Var(\alpha_j) = \emptyset$. d-DNNF is the subset of DNNF where every $\vee$-node $\vee(\alpha_1, \ldots, \alpha_k)$ is deterministic, which means that $\forall i, j \in \{1, \ldots, k\}$, if $i \neq j$ then $\alpha_i \wedge \alpha_j$ is inconsistent. BDD is the subset of C-DAG with $C = \{ite\}$ which consists of DAGs α such that every leaf node is labelled by a Boolean constant, $\top$ or $\bot$. *ite* is a ternary connective ("ite" stands for "if ... then ... else ..."). Usually, instead of labeling a decision node $N = \langle x, N_+, N_- \rangle$ of a BDD formula by the name of the connective used (i.e., "ite") and considering three children for it

[2] The algorithm consists in labeling each node N of α by a set of variables V_N; the nodes are considered in inverse topological ordering, $V_N = var(l)$ when N is a leaf node labeled by l, $V_N = V_M \setminus \{x\}$ when N is an internal node labeled by $\exists x$ or $\forall x$ and M is the child of N, $V_N = \bigcup_{M_i \text{ child of } V_N} V_{M_i}$ when N is an internal node labeled by a connective $c \in C$; $Var(\alpha)$ is equal to V_{N_α} where N_α is the root of α.

(one for x, one for N_+ and one for N_-), N is labelled by x and has only two children (one for N_+ and one for N_-). Given a total, strict ordering $<$ over PS, the class $\mathtt{OBDD}_<$ is the subset of $\mathtt{BDD}$ which consists of DAGs α such that every path from the root of α to a leaf node is compatible with $<$.

As usual, a clause (resp. a term) is a finite disjunction (resp. conjunction) of literals. $\mathtt{CLAUSE}$ is the subset of $\mathtt{DAG\text{-}NNF}$ consisting of all clauses, and $\mathtt{TERM}$ is the subset of $\mathtt{DAG\text{-}NNF}$ consisting of all terms. $\mathtt{NNF}$ is the subset of $\mathtt{DAG\text{-}NNF}$ consisting of formulae (i.e., tree-shaped representations). $\mathtt{CNF}$ is the subset of $\mathtt{NNF}$ consisting of all conjunctions of clauses, while $\mathtt{DNF}$ is the subset of $\mathtt{NNF}$ consisting of all disjunctions of terms. $\mathtt{PI}$ is the subset of $\mathtt{CNF}$ consisting of prime implicates formulae (also known as Blake formulae); a $\mathtt{PI}$ formula is a $\mathtt{CNF}$ formula, the conjunction of all clauses from the set $PI(\alpha)$ for some C-$\mathtt{QDAG}$ representation α; $PI(\alpha)$ contains the prime implicates of α, i.e., the logically strongest clauses which are implied by α (one representative per equivalence class is considered, only). An *essential* prime implicate of α is a prime implicate δ of α such that if the clause equivalent to δ is removed from $PI(\alpha)$, the conjunction of the clauses from the resulting set is no longer equivalent to α. For instance, if $\alpha = (p \Rightarrow q) \wedge (q \Rightarrow r) \wedge (p \Rightarrow (r \vee s))$, then $PI(\alpha) = \{\neg p \vee q, \neg q \vee r, \neg p \vee r\}$. $\neg p \vee q$ and $\neg q \vee r$ are essential prime implicates of α, while $\neg p \vee r$ is not. An important point is that any $\mathtt{CNF}$ formula equivalent to a propositional representation α contains (up to logical equivalence) every essential prime implicate of α.

For space reasons, we do not provide hereafter the definitions of the propositional classes of representations $\mathtt{DNNF}_\mathtt{T}$ and $\mathtt{IP}$ (see [1,18] for formal definitions).

2.2. Semantics

Let us recall that an interpretation (or world) over $V \subseteq PS$ is a mapping ω from V to $BOOL = \{0, 1\}$. Interpretations are sometimes viewed as subsets of PS, consisting of all the variables that are set to 1 by the interpretations. When a total, strict ordering $<$ over PS is considered, the restriction of an interpretation ω over a finite subset $\{x_1, \ldots, x_n\}$ of PS can also be represented as a bit vector; for instance, the restriction of ω over $\{a, b, c\}$ such that $\omega(a) = 1$, $\omega(b) = 0$, and $\omega(c) = 0$ can be represented as 100 when a, b, c are such that $a < b < c$. For any $x \in V$, ω_{-x} is the interpretation over V which coincides with ω on every variable of V, except on x; formally, $\omega_{-x}(y) = \omega(y)$ if $y \neq x$, $= 1 - \omega(x)$ if $y = x$.

We are now ready to define the semantics of C-$\mathtt{QDAG}$ representations in an interpretation ω over PS:

Definition 2 *(Semantics of C-$\mathtt{QDAG}$ representations).* The *semantics* of a C-$\mathtt{QDAG}$ representation α in an interpretation ω over PS is the truth value $[\![\alpha]\!](\omega)$ from $BOOL$ defined inductively as follows:

- If $\alpha = \top$, then $[\![\alpha]\!](\omega) = 1$.
- If $\alpha = \bot$, then $[\![\alpha]\!](\omega) = 0$.
- If α is a positive literal x, then $[\![\alpha]\!](\omega) = \omega(x)$.
- If α is a negative literal $\neg x$, then $[\![\alpha]\!](\omega) = 1 - \omega(x)$.
- If $\alpha = c(\beta_1, \ldots, \beta_n)$, where $c \in C$ has arity n, then $[\![\alpha]\!](\omega) = [\![c]\!]([\![\beta_1]\!](\omega), \ldots, [\![\beta_n]\!](\omega))$, where $[\![c]\!]$ is the Boolean function from $BOOL^n$ to $BOOL$, which is the semantics of c.
- If $\alpha = \exists x.\beta$, then $[\![\alpha]\!](\omega) = 1$ iff $[\![\beta]\!](\omega) = 1$ or $[\![\beta]\!](\omega_{-x}) = 1$.
- If $\alpha = \forall x.\beta$, then $[\![\alpha]\!](\omega) = 1$ iff $[\![\beta]\!](\omega) = 1$ and $[\![\beta]\!](\omega_{-x}) = 1$.

An interpretation ω over PS is said to be a *model* of $\alpha \in C$-$\mathtt{QDAG}$, noted $\omega \models \alpha$, if and only if $[\![\alpha]\!](\omega) = 1$. If α has a model, then it is *consistent*; if every interpretation over PS is a model of α, then α is *valid*. If every model of α is a model of $\beta \in C$-$\mathtt{QDAG}$, then β is a *logical consequence* of α, noted $\alpha \models \beta$. $Mod(\alpha)$ denotes the set of models of α over $Var(\alpha)$. Furthermore, when both $\alpha \models \beta$ and $\beta \models \alpha$ hold, α and β are *logically equivalent*, noted $\alpha \equiv \beta$.

For instance, with $C = \{\wedge, \vee, \neg, \oplus\}$, the C-$\mathtt{QDAG}$ representation given in Fig. 1 is equivalent to the C-$\mathtt{QDAG}$ formula given in Fig. 2.

By structural induction one can easily show that the semantics of any C-$\mathtt{QDAG}$ representation α depends only on its free variables, in the sense that, for any interpretation ω' over PS which coincides with a given interpretation ω on all the free variables of α, ω is a model of α if and only if ω' is a model of α. Accordingly, the semantics of a C-$\mathtt{QDAG}$ representation α in an interpretation ω over PS is fully determined by α and the restriction of ω over $Var(\alpha)$.

Clearly enough, renaming at the same time a quantified occurrence of a variable x in a quantification $\exists x$ or $\forall x$ occurring in a C-$\mathtt{QDAG}$ formula α, and every occurrence of x in α which depends on the quantification leads to a C-$\mathtt{QDAG}$ formula equivalent to α. Furthermore, such a renaming process can be achieved in time linear in the size of α.

However, things are much more tricky when general C-$\mathtt{QDAG}$ representations (not reduced to formulae) are considered. Consider for instance the quantification $\exists q$ occurring in the C-$\mathtt{QDAG}$ representation α reported at Fig. 1, where $C = \{\wedge, \vee, \neg, \oplus\}$. The occurrence of variable q in the leaf of α labelled with literal q depends on this quantification. Replacing q by the fresh variable s in $\exists q$ and at this occurrence would not lead to a representation equivalent to α since s would be a free variable of the resulting representation. Indeed, there exist four paths from the root of α to that leaf, and three of them do not contain any quantified occurrence of q. This is salient on the C-$\mathtt{QDAG}$ formula equivalent to α reported at Fig. 2, and obtained by "unfolding" α. Thus, when some variable occurrence can be both free and bound, renaming quantified variables while preserving equivalence can be a computationally demanding task (the unfolding process may easily lead to an exponential blow-up of the input representation). Actually, when $C \supseteq \{\wedge, \vee\}$, the possibility of having some variable

occurrences both free and bound (or to depend on different existential quantifications) in C-QDAG representations not containing universal quantifications is enough to simulate universal quantifications in them (see [27]). As a consequence, the corresponding class of DAGs is strictly more succinct than the corresponding language of formulae. On the other hand, some problems are computationally easier when formulae (and not DAGs) are considered; for instance, when universal quantifications are disabled, the model checking problem for C-QDAG formulae with $C \supseteq \{\wedge, \vee\}$ is "only" NP-complete, while it is PSPACE-complete when the full class of C-QDAG representations without universal quantifications is considered.

Conventionally, the representation α_N rooted at a decision node $N = \langle x, N_+, N_- \rangle$ over $x \in PS$ in the standard representation of an ordered binary decision diagram (i.e., an $\mathtt{OBDD}_<$ representation) is such that $\alpha_N \equiv ite(x, \alpha_{N_+}, \alpha_{N_-}) \equiv (x \wedge \alpha_{N_+}) \vee (\neg x \wedge \alpha_{N_-})$. α_{N_+} (resp. α_{N_-}), the representation associated with node N_+ (resp. N_-), is the conditioning of α by x (resp. $\neg x$), i.e., the representation obtained by replacing every occurrence of x in α_N by $\top$ (resp. $\bot$).

Finally, we consider the following notations. If $\alpha \in C$-QDAG and $X = \{x_1, \ldots, x_n\} \subseteq PS$, then $\exists X.\alpha$ is a short for

$$\exists x_1.\big(\exists x_2.(\ldots \exists x_n.\alpha)\ldots\big)$$

and $\forall X.\alpha$ is a short for

$$\forall x_1.\big(\forall x_2.(\ldots \forall x_n.\alpha)\ldots\big)$$

(these notations are well-founded since whatever the chosen ordering on X, the resulting representations are logically equivalent).

2.3. KROM, HORN, AFF, K/H, *and* renH

In the following, we will focus on several well-known propositional languages, namely the Krom CNF language KROM (also known as the bijunctive fragment) [23], the Horn CNF language HORN [24], and the affine language AFF (also known as the biconditional fragment) [25], as well as K/H (Krom or Horn CNF formulae) and renH, the language of renamable Horn CNF formulae [26].

The languages KROM, HORN, AFF, K/H, and renH are formally defined as follows:

Definition 3 *(*KROM, HORN, AFF, K/H, *and* renH*).*

- The language KROM is the subset of all CNF formulae in which each clause is binary, i.e., it contains at most two literals.
- The language HORN is the subset of all CNF formulae in which each clause is Horn, i.e., it contains at most one positive literal.
- The language K/H is the union of KROM and HORN.
- The language renH is the subset of all CNF formulae α for which there exists a subset V of $Var(\alpha)$ (called a Horn renaming for α) such that the formula noted $V(\alpha)$ obtained by substituting in α every literal l of L_V by its complementary literal $\bar{l}$ is a HORN formula.
- The language AFF is the subset of C-DAG with $C = \{\wedge, \neg, \oplus\}$, consisting of conjunctions of XOR-clauses where a XOR-clause is a finite XOR-disjunction of literals (the XOR connective is denoted by $\oplus$).

Here are some examples of formulae from KROM, HORN, renH, and AFF:

- $(x \vee y) \wedge (\neg y \vee z)$ is a KROM formula.
- $(\neg x \vee \neg y \vee z) \wedge (\neg y \vee z)$ is a HORN formula.
- $(x \vee y \vee z) \wedge (\neg x \vee \neg y \vee z)$ is a renH formula.
  $V = \{x, z\}$ is a Horn renaming for it.
- $(x \oplus \neg y \oplus \top) \wedge (\neg x \oplus y \oplus z)$ is an AFF formula.

Clearly enough, determining whether a given C-QDAG representation α (for any fixed C) is a KROM (resp. HORN, K/H, AFF) formula can be easily achieved in time polynomial in the size of α. Note also that there exists linear time algorithms for recognizing renH formulae (see e.g. [28,29]); furthermore, such recognition algorithms typically give a Horn renaming when it exists.

KROM, HORN, AFF, K/H, and renH are known as polynomial classes for the SAT problem (i.e., the restriction of SAT to any of them is in polynomial time – stated otherwise, each of them satisfies **CO**). However, none of them is fully expressive w.r.t. propositional logic (there exist propositional formulae, like $(x \vee y \vee z) \wedge (\neg x \vee \neg y \vee \neg z)$, which cannot be represented in any of them); this severely restricts their attractiveness for the KC purpose.

Interestingly, KROM, HORN, and AFF have semantical characterizations in terms of closures of sets of models:

- A set S of interpretations over a finite $V \subseteq PS$ is the set of models of a KROM formula α such that $Var(\alpha) = V$ if and only if it is closed for majority [30,25], i.e., $\forall \omega_1, \omega_2, \omega_3 \in S$, the interpretation $maj(\omega_1, \omega_2, \omega_3)$ over V belongs to S as well. Here $maj(\omega_1, \omega_2, \omega_3)$ is defined by $\forall x \in V$, $maj(\omega_1, \omega_2, \omega_3)(x) = 1$ if at least two interpretations among $\omega_1, \omega_2, \omega_3$ give the truth value 1 to x.

- A set S of interpretations over a finite $V \subseteq PS$ is the set of models of a HORN formula α such that $Var(\alpha) = V$ if and only if it is closed for the bitwise AND connective [31,32], i.e., $\forall \omega_1, \omega_2 \in S$, the interpretation $and(\omega_1, \omega_2)$ over V belongs to S. Here $and(\omega_1, \omega_2)$ is defined by $\forall x \in V$, $and(\omega_1, \omega_2)(x) = 1$ if $\omega_1(x) = \omega_2(x) = 1$.
- A set S of interpretations over a finite $V \subseteq PS$ is the set of models of an AFF formula α such that $Var(\alpha) = V$ if and only if S is closed for the ternary $\oplus$ connective [30,25], i.e., $\forall \omega_1, \omega_2, \omega_3 \in S$, the interpretation $\oplus(\omega_1, \omega_2, \omega_3)$ over V belongs to S. Here $\oplus(\omega_1, \omega_2, \omega_3)$ is defined by $\forall x \in V$, $\oplus(\omega_1, \omega_2, \omega_3)(x) = \omega_1(x) \oplus \omega_2(x) \oplus \omega_3(x)$.

These characterization results can be exploited to show that some propositional formulae cannot be expressed as KROM (resp. HORN, AFF) formulae.

3. Queries, transformations, expressiveness and succinctness

Let us now briefly recall the sets of queries and transformations used for comparing propositional languages in [1], as well as the notions of expressiveness and succinctness; their importance is discussed in depth in [1], so we refrain from recalling it here.

3.1. Queries

The basic queries considered in [1] and subsequent papers concern DAG-NNF representations; they include tests for consistency (**CO**), validity (**VA**), clausal entailment (**CE**), implicants (**IM**), equivalence (**EQ**), sentential entailment (**SE**), counting (**CT**) and enumerating theory models (**ME**). We extend them to C-QDAG representations and add to them **MC**, the model checking query, which is trivially offered by unquantified representations, but not by quantified representations in the general case.

Definition 4 *(Queries).* Let $\mathcal{L}$ denote any subset of C-QDAG.

- $\mathcal{L}$ satisfies **CO**, the *consistency* query (resp. **VA**, the *validity* query) if there exists a polynomial-time algorithm that maps every representation α from $\mathcal{L}$ to 1 if α is consistent (resp. valid), and to 0 otherwise.
- $\mathcal{L}$ satisfies **CE**, the *clausal entailment* query, if there exists a polynomial-time algorithm that maps every pair $\langle \alpha, \delta \rangle$, where α is a representation from $\mathcal{L}$ and δ is a clause, to 1 if $\alpha \models \delta$ holds, and to 0 otherwise.
- $\mathcal{L}$ satisfies **EQ**, the *equivalence* query (resp. **SE**, the *sentential entailment* query) if there exists a polynomial-time algorithm that maps every pair $\langle \alpha, \beta \rangle$ of representations from $\mathcal{L}$ to 1 if $\alpha \equiv \beta$ (resp. $\alpha \models \beta$) holds, and to 0 otherwise.
- $\mathcal{L}$ satisfies **IM**, the *implicant* query, if there exists a polynomial-time algorithm that maps every pair $\langle \alpha, \gamma \rangle$, where α is a representation from $\mathcal{L}$ and γ is a term, to 1 if $\gamma \models \alpha$ holds, and to 0 otherwise.
- $\mathcal{L}$ satisfies **CT**, the *model counting* query, if there exists a polynomial-time algorithm that maps every representation α from $\mathcal{L}$ to a nonnegative integer that represents the number of models of α over $Var(\alpha)$ (in binary notation).
- $\mathcal{L}$ satisfies **ME**, the *model enumeration* query, if there exists a polynomial $p(.,)$. and an algorithm that outputs all models of an arbitrary representation α from $\mathcal{L}$ in time $p(n, m)$, where n is the size of α and m is the number of its models over $Var(\alpha)$.
- $\mathcal{L}$ satisfies **MC**, the *model checking* query, if there exists a polynomial-time algorithm that maps every pair $\langle \alpha, \omega \rangle$, where α is a representation from $\mathcal{L}$ and ω is an interpretation over $Var(\alpha)$, to 1 if I is a model of α, and to 0 otherwise.

3.2. Transformations

The basic transformations considered in [1] are conditioning (**CD**), (possibly bounded) closures under the connectives[3] $\wedge$, $\vee$, and $\neg$ ($\wedge$**C**, $\wedge$**BC**, $\vee$**C**, $\vee$**BC**, $\neg$**C**) and (possibly bounded) forgetting which can be viewed as a closure operation under existential quantification (**FO**, **SFO**). Forgetting is an important transformation as it allows us to focus/project a representation on a set of variables, which proves helpful in many applications, including model-based diagnosis [33], reasoning about actions [34], and reasoning under inconsistency [35,36]. All those transformations concern DAG-NNF representations. We extend them to C-QDAG representations and enrich the list with two additional transformations, which are dual to (**FO**, **SFO**), namely "ensuring" (**EN**) and the bounded restriction of it (**SEN**). Ensuring amounts to eliminating universal quantifications and allows us to project a representation on a set of variables in a robust way, i.e., independently of the values of the removed variables. This transformation is central in decision making under uncertainty and non-deterministic planning, see e.g. [37].

[3] Closures under other connectives could also be easily defined but seem to be less significant.

Definition 5 *(Transformations).* Let $\mathcal{L}$ denote any subset of C-QDAG.

- $\mathcal{L}$ satisfies **CD**, the *conditioning* transformation, if there exists a polynomial-time algorithm that maps every pair $\langle \alpha, \gamma \rangle$, where α is a representation from $\mathcal{L}$ and γ is a consistent term, to a representation from $\mathcal{L}$ that is logically equivalent to $\exists Var(\gamma).(\alpha \wedge \gamma)$.
- $\mathcal{L}$ satisfies **FO**, the *forgetting* transformation, if there exists a polynomial-time algorithm that maps every pair $\langle \alpha, X \rangle$, where α is a representation from $\mathcal{L}$ and X is a set of variables from PS, to a representation from $\mathcal{L}$ equivalent to $\exists \mathbf{X}.\alpha$. If the property holds for each singleton $\mathbf{X}$, we say that $\mathcal{L}$ satisfies **SFO** (singleton forgetting).
- $\mathcal{L}$ satisfies **EN**, the *ensuring* transformation, if there exists a polynomial-time algorithm that maps every pair $\langle \alpha, X \rangle$, where α is a representation from $\mathcal{L}$ and X is a set of variables from PS, to a representation from $\mathcal{L}$ equivalent to $\forall \mathbf{X}.\alpha$. If the property holds for each singleton $\mathbf{X}$, we say that $\mathcal{L}$ satisfies **SEN** (singleton ensuring).
- $\mathcal{L}$ satisfies $\wedge$**C**, the *closure under conjunction* transformation (resp. $\vee$**C**, the *closure under disjunction* transformation) if there exists a polynomial-time algorithm that maps every finite set of representations $\alpha_1, \ldots, \alpha_n$ from $\mathcal{L}$ to a representation of $\mathcal{L}$ that is equivalent to $\alpha_1 \wedge \ldots \wedge \alpha_n$ (resp. $\alpha_1 \vee \ldots \vee \alpha_n$).
- $\mathcal{L}$ satisfies $\wedge$**BC**, the *bounded closure under conjunction* transformation (resp. $\vee$**BC**, the *bounded closure under disjunction* transformation), if there exists a polynomial-time algorithm that maps every pair of representations α and β from $\mathcal{L}$ to a representation of $\mathcal{L}$ that is equivalent to $\alpha \wedge \beta$ (resp. $\alpha \vee \beta$).
- $\mathcal{L}$ satisfies $\neg$**C**, the *closure under negation* transformation, if there exists a polynomial-time algorithm that maps every representation α from $\mathcal{L}$ to a representation of $\mathcal{L}$ which is equivalent to $\neg\alpha$.

When α is a C-DAG representation (i.e., a non-quantified representation), the conditioning of α by γ can be defined in an equivalent, yet simpler way, as the representation $\alpha_{|\gamma}$ obtained by replacing in α every occurrence of variable x by $\top$ (resp. $\bot$) when x (resp. $\neg x$) is a literal of γ. Such a characterization cannot be extended to C-QDAG representations in the general case. Especially, considering only those variables x occurring free in α as candidates for the replacement is not enough. Indeed, since DAG-based representations are considered, it can be the case that in α one can find a leaf node N labeled by x such that one path from the root of α to this leaf node does not contain any node labeled by a quantification on x, while other paths from the root to N contain such quantifications (see [27]).

3.3. Expressiveness, succinctness, and polynomial translations

We consider three notions of translations on classes of propositional representations (here, subsets of C-QDAG), starting from the less demanding one, namely expressiveness:

Definition 6 *(Expressiveness).* Let $\mathcal{L}_1$ and $\mathcal{L}_2$ be two subsets of C-QDAG. $\mathcal{L}_1$ is *at least as expressive as* $\mathcal{L}_2$, denoted $\mathcal{L}_1 \leq_e \mathcal{L}_2$, if for every representation $\alpha \in \mathcal{L}_2$, there exists an equivalent representation $\beta \in \mathcal{L}_1$.

A first refinement of such a notion of translatability consists in considering only *polynomial-space translations*, i.e., the size of the translated representation must remain polynomial in the size of the input representation:

Definition 7 *(Succinctness).* Let $\mathcal{L}_1$ and $\mathcal{L}_2$ be two subsets of C-QDAG. $\mathcal{L}_1$ is *at least as succinct as* $\mathcal{L}_2$, denoted $\mathcal{L}_1 \leq_s \mathcal{L}_2$, if there exists a polynomial p such that for every representation $\alpha \in \mathcal{L}_2$, there exists an equivalent representation $\beta \in \mathcal{L}_1$ where $|\beta| \leq p(|\alpha|)$.

Finally, we consider still more demanding translations, namely *polynomial-time translations*:

Definition 8 *(Polynomial translation).* Let $\mathcal{L}_1$ and $\mathcal{L}_2$ be two subsets of C-QDAG. $\mathcal{L}_2$ is said to be *polynomially translatable* into $\mathcal{L}_1$, noted $\mathcal{L}_1 \leq_p \mathcal{L}_2$, if there exists a (deterministic) polynomial-time algorithm f such that for every $\alpha \in \mathcal{L}_2$, we have $f(\alpha) \in \mathcal{L}_1$ and $f(\alpha) \equiv \alpha$. We also say that α is polynomially translatable into $f(\alpha)$.

Clearly enough, $\leq_e$, $\leq_s$, and $\leq_p$ are pre-orders (i.e., reflexive and transitive relations) over the subsets of C-QDAG. Furthermore, we have the inclusions:

$$\leq_p \subset \leq_s \subset \leq_e$$

For each relation $\leq_*$ among $\leq_e$, $\leq_s$, and $\leq_p$, the relation $\sim_*$ denotes the symmetric part of $\leq_*$, defined by $\mathcal{L}_1 \sim_* \mathcal{L}_2$ if $\mathcal{L}_1 \leq_* \mathcal{L}_2$ and $\mathcal{L}_2 \leq_* \mathcal{L}_1$. By construction, each $\sim_*$ is an equivalence relation (i.e., a reflexive, symmetric and transitive relation). On the other hand,the relation $<_*$ denotes the asymmetric part of $\leq_*$, defined by $\mathcal{L}_1 <_* \mathcal{L}_2$ if $\mathcal{L}_1 \leq_* \mathcal{L}_2$ and $\mathcal{L}_2 \not\leq_* \mathcal{L}_1$. By construction, each $<_*$ is a strict order (i.e., an irreflexive and transitive relation).

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 (which is considered very unlikely in complexity theory).

When $\mathcal{L}_1 \leq_e \mathcal{L}_2$ holds, every representation from $\mathcal{L}_2$ can be translated into an equivalent representation from $\mathcal{L}_1$. The minimal elements w.r.t. $\leq_e$ (i.e., the most expressive elements) of the set of all subsets of C-QDAG when C is any functionally complete set of connectives (especially, as soon as C contains $\vee$ and $\wedge$ since leaf nodes of C-QDAG representations are labeled by literals) are called *complete* propositional classes: they can provide a representation (up to logical equivalence) of any Boolean function.

When $\mathcal{L}_1 \sim_e \mathcal{L}_2$ (resp. $\mathcal{L}_1 \sim_s \mathcal{L}_2$, $\mathcal{L}_1 \sim_p \mathcal{L}_2$), $\mathcal{L}_1$ and $\mathcal{L}_2$ are said to be equally expressive (resp. equally succinct, polynomially equivalent).

Whenever $\mathcal{L}_1$ is polynomially translatable into $\mathcal{L}_2$, every query which can be answered in polynomial time in $\mathcal{L}_2$ can also be answered in polynomial time in $\mathcal{L}_1$; and conversely, every query which cannot be answered in polynomial time in $\mathcal{L}_1$ unless $\mathsf{P} = \mathsf{NP}$ cannot be answered in polynomial time in $\mathcal{L}_2$, unless $\mathsf{P} = \mathsf{NP}$. Furthermore, polynomially equivalent classes are equally efficient in the sense that they possess the same set of tractable queries and transformations.

4. On closures of propositional representations

Intuitively, a closure principle applied to a class $\mathcal{L}$ of propositional representations defines a new class, called a closure of $\mathcal{L}$, through the (implicit) application of "operators" (i.e., connectives from C or quantifications) to the representations from $\mathcal{L}$. Formally:

Definition 9 *(Closure).* Let $\mathcal{L} \subseteq C$-QDAG and $\triangle \subseteq C \cup \{\forall, \exists\}$. The *closure* $\mathcal{L}[\triangle]$ of $\mathcal{L}$ by $\triangle$ is the subset of C-QDAG inductively defined as follows[4]:

1. if $\alpha \in \mathcal{L}$, then $\alpha \in \mathcal{L}[\triangle]$,
2. if $c \in \triangle$ is an n-ary connective and $\alpha_1, \ldots, \alpha_n$ are elements of $\mathcal{L}[\triangle]$ such that $\forall i, j \in \{1, \ldots, n\}$, if $i \neq j$ then α_i and α_j do not share any common (nonempty) subgraphs, then $c(\alpha_1, \ldots, \alpha_n) \in \mathcal{L}[\triangle]$,
3. if $c \in \triangle$ is a quantifier, $x \in PS$, and $\alpha \in \mathcal{L}[\triangle]$, then $cx.\alpha \in \mathcal{L}[\triangle]$.

Each element of $\mathcal{L}[\triangle]$ can be viewed as a "tree" which "internal nodes" are labeled by connectives from C or quantifications and its "leaf nodes" correspond to "independent" representations from $\mathcal{L}$. Accordingly, the representations α_i considered in item 2. of Definition 9 do not share any common subgraphs.

Clearly, if there exists a polynomial-time algorithm for determining whether a given representation $\alpha \in C$-QDAG belongs or not to $\mathcal{L}$, then there also exists a polynomial-time algorithm for determining whether a given representation $\alpha \in C$-QDAG belongs or not to the closure $\mathcal{L}[\triangle]$ of $\mathcal{L}$ by $\triangle$.

We have derived the following (easy) proposition, which rules the inclusions between closures depending on the way their sets of connectives are related by set inclusion:

Proposition 1. *For every subset $\mathcal{L}$, $\mathcal{L}'$ of C-QDAG and every subset $\triangle_1$, $\triangle_2$ of $C \cup \{\exists, \forall\}$, we have:*

0. $\mathcal{L} \subseteq \mathcal{L}[\triangle_1]$, *and if* $\mathcal{L} \subseteq \mathcal{L}'$, *then* $\mathcal{L}[\triangle_1] \subseteq \mathcal{L}'[\triangle_1]$.
1. $(\mathcal{L}[\triangle_1])[\triangle_2] \subseteq \mathcal{L}[\triangle_1 \cup \triangle_2]$.
2. $(\mathcal{L}[\triangle_1])[\triangle_1] = \mathcal{L}[\triangle_1]$.
3. *If* $\triangle_1 \subseteq \triangle_2$ *then* $\mathcal{L}[\triangle_1] \subseteq \mathcal{L}[\triangle_2]$.
4. *If* $\triangle_1 \subseteq \triangle_2$ *then* $(\mathcal{L}[\triangle_1])[\triangle_2] = \mathcal{L}[\triangle_2]$ *and* $(\mathcal{L}[\triangle_2])[\triangle_1] = \mathcal{L}[\triangle_2]$.

Some additional properties stating how some closures of a class $\mathcal{L}$ can be composed, can be derived when bound variables can be "freely" renamed in the $\mathcal{L}$ representations. The property of stability by uniform renaming, given at Definition 10, characterizes the subsets of C-QDAG for which, intuitively, the choice of variable names in the $\mathcal{L}$ representations does not really matter:

Definition 10 *(Stability by uniform renaming).* Let $\mathcal{L}$ be any subset of C-QDAG. $\mathcal{L}$ is *stable by uniform renaming* if for every $\alpha \in \mathcal{L}$, for every non-empty subset V of variables occurring in α, there exist arbitrarily many distinct bijections r_i ($i \in \mathbb{N}$) from V to subsets V_i of fresh variables from PS (i.e., for each $i, j \in \mathbb{N}$ with $i \neq j$, we have $V_i \cap V_j = V_i \cap V = \emptyset$) such that the representation $r_i(\alpha)$ obtained by replacing in α (in a uniform way) every occurrence of $x \in V$ (either quantified or non-quantified) by $r_i(x)$ belongs to $\mathcal{L}$ as well.

This condition is not very demanding: all the "standard" classes of propositional representations (quantified or not) are stable by uniform renaming (when based on a countably infinite set PS as this is the case here). Special attention must nevertheless be paid to the $\mathrm{OBDD}_<$ language, and more generally to every class based on an ordered set of propositional

[4] In order to alleviate the notations, when $\triangle = \{\delta_1, \ldots, \delta_n\}$, we write $\mathcal{L}[\delta_1, \ldots, \delta_n]$ instead of $\mathcal{L}[\{\delta_1, \ldots, \delta_n\}]$.

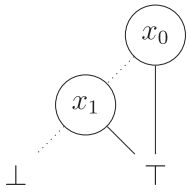

Fig. 3. An $\mathtt{OBDD}_<$ representation of $x_0 \vee x_1$.

variables. For the $\mathtt{OBDD}_<$ case where $<$ is a strict and complete ordering over PS we may assume the ordered set $(PS, <)$ to be of order type η (η is the order type of the set of rational numbers equipped with its usual ordering [38]). This restriction is harmless since the set of variables occurring in any $\mathtt{OBDD}_<$ representation is finite. In a nutshell, whatever the way the variables occurring in a given $\mathtt{OBDD}_<$ representation α are ordered w.r.t. $<$, one must be able to "insert" in this ordering arbitrarily many fresh variables between two variables of α while preserving the way other variables are ordered. Order type η clearly allows it (between two distinct rational numbers one can find countably many rationals). To make things clearer, let us give a counter-example: let $PS = \{x_i \mid i \in \mathbb{N}\}$ ordered in such a way that for every $i \in \mathbb{N}$, $x_i < x_{i+1}$. Consider an $\mathtt{OBDD}_<$ representation of $x_0 \vee x_1$ as given in Fig. 3. $<$ is not of type η. Take $V = \{x_0\}$: x_0 cannot be renamed into a different variable from PS without questioning the ordering requirement over $\mathtt{OBDD}_<$, which shows that $\mathtt{OBDD}_<$ is not stable by uniform renaming in this case.

Straightforwardly, the closure by any set of connectives/quantifiers of any class of propositional representations, which is stable by uniform renaming, also is stable by uniform renaming.

We are now ready to present more specific results. The following polynomial (dual) equivalences, showing that existential quantifications (resp. universal quantifications) when viewed as "operators" "distribute" over disjunctions (resp. conjunctions), are well-known:

$$\exists x.(\alpha_1 \vee \ldots \vee \alpha_n) \equiv (\exists x.\alpha_1) \vee \ldots \vee (\exists x.\alpha_n),$$

$$\forall x.(\alpha_1 \wedge \ldots \wedge \alpha_n) \equiv (\forall x.\alpha_1) \wedge \ldots \wedge (\forall x.\alpha_n).$$

It can then be shown that:

Proposition 2. *Let $\mathcal{L}$ be any subset of C-$\mathtt{QDAG}$ s.t. $\mathcal{L}$ is stable by uniform renaming. We have:*

- $(\mathcal{L}[\exists])[\vee] \sim_p (\mathcal{L}[\vee])[\exists] \sim_p \mathcal{L}[\vee, \exists]$.
- $(\mathcal{L}[\forall])[\wedge] \sim_p (\mathcal{L}[\wedge])[\forall] \sim_p \mathcal{L}[\wedge, \forall]$.

Fig. 4 illustrates the polynomial equivalences between disjunctive closures given at Proposition 2.

Proposition 1 and Proposition 2 show together that when $\Delta = \{\vee, \exists\}$ (resp. $\{\wedge, \forall\}$) closing $\mathcal{L}[\Delta]$ by subsets of Δ in an iterative fashion does not lead to a "new" class, i.e., a class which is not polynomially equivalent to $\mathcal{L}$. Especially, we have

$$\big(\mathcal{L}[\vee, \exists]\big)[\exists] \sim_p \big(\mathcal{L}[\vee, \exists]\big)[\vee] \sim_p \mathcal{L}[\vee, \exists].$$

This shows, so to say, that the "sequential" closure of a propositional class, stable by uniform renaming, by a set of operators among $\{\vee, \exists\}$ (resp. among $\{\wedge, \forall\}$) is polynomially equivalent to its "parallel" closure. No similar result can be systematically guaranteed for arbitrary choices of classes and operators. For instance, if $\mathcal{L}$ is the set of literals over PS, then the "sequential" closure $(\mathcal{L}[\vee])[\wedge]$ is the set of all $\mathtt{CNF}$ formulae, the "sequential" closure $(\mathcal{L}[\wedge])[\vee]$ is the set of all $\mathtt{DNF}$ formulae, and the "parallel" closure $\mathcal{L}[\vee, \wedge]$ is the set of all $\mathtt{NNF}$ representations. It is well-known that those three languages are not pairwise polynomially equivalent (indeed, we have $\mathtt{CNF} \not\leq_s \mathtt{DNF}$, $\mathtt{DNF} \not\leq_s \mathtt{CNF}$, $\mathtt{NNF} <_s \mathtt{CNF}$, and $\mathtt{NNF} <_s \mathtt{DNF}$, see e.g. [39]). Similarly, if $\mathcal{L} = \mathtt{CLAUSE}$, then $(\mathcal{L}[\wedge])[\exists]$ and $\mathcal{L}[\wedge, \exists]$ are polynomially equivalent to $\mathtt{CNF}[\exists]$, but $(\mathcal{L}[\exists])[\wedge]$ is polynomially equivalent to $\mathtt{CNF}$, which is not polynomially equivalent to $\mathtt{CNF}[\exists]$. Indeed, whatever C, C-$\mathtt{DAG}$ is polynomially translatable into $\mathtt{CNF}[\exists]$ using Tseitin's extension principle [40], while $\mathtt{CNF}$ is not at least as succinct as C-$\mathtt{DAG}$ as soon as $C \supseteq \{\wedge, \vee, \neg\}$ (indeed, $\mathtt{CNF}$ is not at least as succinct as the subset $\mathtt{DNF}$ of $\mathtt{NNF}$, which is itself a subset of C-$\mathtt{DAG}$ in this case).

We have derived the following proposition, which relates the queries and the transformations offered by $\mathcal{L}$, with the queries and transformations offered by its disjunctive closures $\mathcal{L}[\vee]$ (the disjunction closure of $\mathcal{L}$), $\mathcal{L}[\exists]$ (the existential closure of $\mathcal{L}$), and $\mathcal{L}[\vee, \exists]$ (the full disjunctive closure of $\mathcal{L}$).

Proposition 3. *Let $\mathcal{L}$ be any subset of C-$\mathtt{QDAG}$ s.t. $\mathcal{L}$ is stable by uniform renaming.*

- *If $\mathcal{L}$ satisfies* **CO** *(resp.* **CD***), then $\mathcal{L}[\vee]$, $\mathcal{L}[\exists]$ and $\mathcal{L}[\vee, \exists]$ satisfy* **CO** *(resp.* **CD***).*
- *If $\mathcal{L}$ satisfies* **CO** *and* **CD***, then $\mathcal{L}$ satisfies* **CE** *and* **ME***.*
- *If $\mathcal{L}$ satisfies* **CO** *and* **CD***, then $\mathcal{L}$, $\mathcal{L}[\vee]$, $\mathcal{L}[\exists]$ and $\mathcal{L}[\vee, \exists]$ satisfy* **MC***.*
- *$\mathcal{L}[\vee]$ and $\mathcal{L}[\vee, \exists]$ satisfy* $\vee$**C** *(hence* $\vee$**BC***) and $\mathcal{L}[\exists]$ and $\mathcal{L}[\vee, \exists]$ satisfy* **FO** *(hence* **SFO***).*

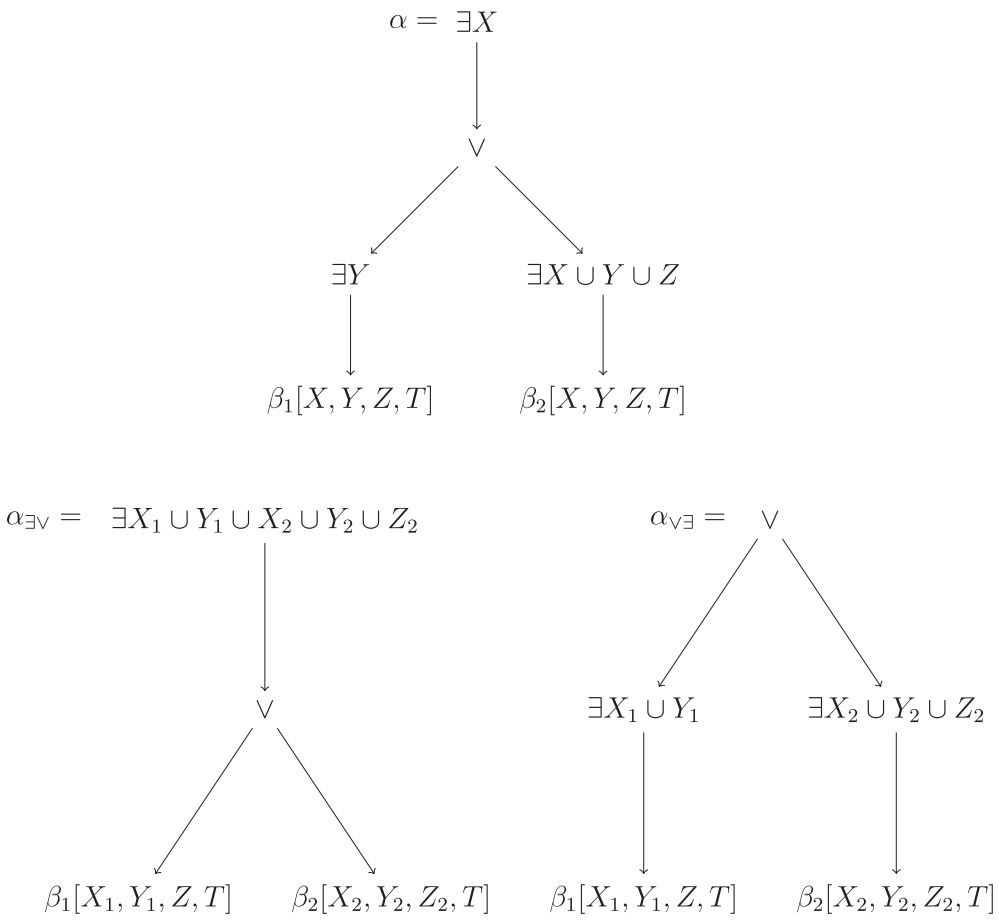

Fig. 4. Polynomial equivalences between disjunctive closures. The representation α at the top of the picture belongs to $\mathcal{L}[\vee, \exists]$. $\beta_1[X, Y, Z, T]$ and $\beta_2[X, Y, Z, T]$ denote propositional representations (not necessarily tree-structured ones) from $\mathcal{L}$ such that $Var(\beta_1) = Var(\beta_2) = X \cup Y \cup Z \cup T$, where X, Y, Z, and T are pairwise disjoint, finite subsets of PS. The representation $\alpha_{\exists\vee}$ at the bottom, left-hand side of the picture is an $(\mathcal{L}[\vee])[\exists]$ representation into which α can be polynomially translated. The representation $\alpha_{\vee\exists}$ at the bottom, right-hand side of the picture is an $(\mathcal{L}[\exists])[\vee]$ representation into which α can be polynomially translated.

- *If $\mathcal{L}$ satisfies* **FO** *(resp.* **SFO***), then $\mathcal{L}[\vee]$ satisfies* **FO** *(resp.* **SFO***).*
- *If $\mathcal{L}$ satisfies* $\wedge$**C** *(resp.* $\wedge$**BC**, $\vee$**C**, $\vee$**BC***), then $\mathcal{L}[\exists]$ satisfies* $\wedge$**C** *(resp.* $\wedge$**BC**, $\vee$**C**, $\vee$**BC***).*

Note that applying disjunctive closures do not preserve other queries or transformations in the general case. Thus:

- If $\mathcal{L}$ satisfies **VA** (resp. **IM**, **CT**, **EQ**, and **SE**), then it can be the case that $\mathcal{L}[\vee]$ does not satisfy it. For instance, TERM satisfies each of **VA**, **IM**, **CT**, **EQ**, and **SE**, but TERM$[\vee]$ = DNF does not satisfy any of them unless P = NP [1].
- If $\mathcal{L}$ satisfies **VA** (resp. **IM**, **CT**, **EQ**, and **SE**), then it can be the case that $\mathcal{L}[\exists]$ does not satisfy it. Thus, $\mathcal{L}$ = CNF satisfies **VA** and **IM** but CNF$[\exists]$ does not satisfy any of them unless P = NP; indeed, DNF (which does not offer any of them) is polynomially translatable into CNF$[\exists]$ using Tseitin's transformation [40]. Similarly, $\mathcal{L}$ = HORN satisfies both **EQ** and **SE**, but HORN$[\exists]$ does not offer any of them (see Proposition 5). Finally, the subset $\mathcal{L}$ = d-DNF of DNF consisting of deterministic DNF formulae (i.e., the DNF formulae $\alpha = \bigvee_{i=1}^{n} \gamma_i$ such that for each $i, j \in 1, \ldots, n$, if $i \neq j$, then the terms γ_i and γ_j are such that $\gamma_i \wedge \gamma_j$ is inconsistent) satisfies **CT**, but d-DNF$[\exists]$ does not. Indeed, DNF is polynomially translatable into d-DNF$[\exists]$: with each DNF formula $\alpha = \bigvee_{i=1}^{n} \gamma_i$ we can associate in polynomial time the equivalent d-DNF$[\exists]$ formula $\exists\{y_1, \ldots, y_n\}.\bigvee_{i=1}^{n}(y_i \wedge \bigwedge_{j=1}^{i-1} \neg y_j \wedge \gamma_i)$, where $\{y_1, \ldots, y_n\}$ is a set of fresh variables (disjoint from $Var(\alpha)$).
- If $\mathcal{L}$ satisfies $\wedge$**C**, then it can be the case that $\mathcal{L}[\vee]$ does not satisfy it. Thus, $\mathcal{L}$ = TERM satisfies $\wedge$**C**, but TERM$[\vee]$ = DNF does not, unless P = NP [1].
- If $\mathcal{L}$ satisfies $\neg$**C**, then it can be the case that none of $\mathcal{L}[\vee]$ and $\mathcal{L}[\exists]$ satisfies it. Thus, $\mathcal{L}$ = OBDD$_<$ satisfies $\neg$**C**, but none of OBDD$_<[\vee]$ and OBDD$_<[\exists]$ satisfies $\neg$**C** unless P = NP. As to OBDD$_<[\vee]$, this comes from the fact that TERM $\geq_p$ OBDD$_<$, which implies that DNF $\geq_p$ OBDD$_<[\vee]$. Since every CNF formula α is polynomially translatable into the negation of a DNF formula β, if OBDD$_<[\vee]$ would satisfy $\neg$**C**, then the consistency of α could be tested in polynomial time by computing first an OBDD$_<[\vee]$ representation equivalent to β, then "negating" it to reach an OBDD$_<[\vee]$ representation equivalent to α. Indeed, since OBDD$_<$ satisfies **CO**, OBDD$_<[\vee]$ also satisfies **CO** (see Proposition 3). As to OBDD$_<[\exists]$, we can make a rather similar proof given that OBDD$_<[\exists]$ also satisfies **CO** (see again Proposition 3). To

get the proof, it is enough to show that $\mathtt{OBDD}_<[\vee] \geq_p \mathtt{OBDD}_<[\exists]$: let $\alpha = \bigvee_{i=1}^{n} \alpha_i$ be an $\mathtt{OBDD}_<[\vee]$ representation; let $y_1, \ldots, y_n$ be variables from $PS \setminus Var(\alpha)$ such that each y_i $(i \in 1, \ldots, n)$ precedes every variable from $Var(\alpha)$. From α, we can generate in polynomial time the $\mathtt{OBDD}_<$ representation

$$\beta = \langle y_1, \alpha_1, \langle y_2, \alpha_2, \ldots, \langle y_n, \alpha_n, \bot, \rangle \ldots \rangle \rangle.$$

To conclude the proof it is enough to observe that α is equivalent to the $\mathtt{OBDD}_<[\exists]$ representation $\exists\{y_1, \ldots, y_n\}.\beta$.

- If $\mathcal{L}$ satisfies **EN**, then it can be the case that $\mathcal{L}[\vee]$ does not satisfy it. Thus, $\mathcal{L} = \mathtt{TERM}$ satisfies **EN**, but $\mathtt{TERM}[\vee] = \mathtt{DNF}$ does not, unless $\mathsf{P} = \mathsf{NP}$, since a DNF formula α is valid iff its universal closure $\forall Var(\alpha).\alpha$ is valid iff $\forall Var(\alpha).\alpha$ is consistent (since $\forall Var(\alpha).\alpha$ has no free variable, it is equivalent to $\top$ or to $\bot$, hence it is consistent precisely when it is valid), and DNF satisfies **CO**.
- If $\mathcal{L}$ satisfies **EN**, then it can be the case that $\mathcal{L}[\exists]$ does not satisfy it. Thus, $\mathcal{L} = \mathtt{CNF}$ satisfies **EN**, but $\mathtt{CNF}[\exists]$ does not, unless PH collapses. This comes easily from the fact that the validity problem for $\mathtt{CNF}[\exists]$ formulae of the form $\exists X.\alpha$ is Π_2^p-complete. Indeed, $\exists X.\alpha$ is valid iff the (closed) quantified Boolean formula $\forall Var(\alpha) \setminus X.(\exists X.\alpha)$ is valid.

5. On the disjunctive closures of KROM, HORN, AFF, K/H, and renH

Let us now focus on the disjunctive closures of KROM, HORN, AFF, K/H, and renH.

First of all, it is obvious that the four languages KROM, HORN, K/H, and AFF are stable by uniform renaming. This is also the case for renH: if V is a Horn renaming for a renH formula α, and if α'_X is the CNF formula obtained by substituting in a uniform way in α every occurrence of a variable v from $X \subseteq PS$ by the fresh variable v', then α'_X also is a renH formula and $V' = \{v \in Var(\alpha') \mid v \in V \setminus X\} \cup \{v' \in Var(\alpha') \mid v \in V \cap X\}$ is a Horn renaming for it.

Now, thanks to Propositions 1 and 2, it is enough to consider the three disjunctive closures $\mathcal{L}[\exists]$, $\mathcal{L}[\vee]$, and $\mathcal{L}[\vee, \exists]$ with $\mathcal{L}$ being any on the five above languages. Clearly enough, the disjunction (resp. existential, full disjunctive) closure of any language among KROM, HORN, K/H, renH, and AFF is also stable by uniform renaming.

Applying the disjunctive closure principles $[\vee]$, $[\exists]$, and $[\vee, \exists]$ to the five languages KROM, HORN, K/H, renH, and AFF leads to consider fifteen additional languages. The following easy result shows that some of the resulting languages do not need to be considered separately, because they are polynomially equivalent.

Proposition 4.

- $\mathtt{KROM} \sim_p \mathtt{KROM}[\exists]$.
- $\mathtt{KROM}[\vee] \sim_p \mathtt{KROM}[\vee, \exists]$.
- $\mathtt{AFF} \sim_p \mathtt{AFF}[\exists]$.
- $\mathtt{AFF}[\vee] \sim_p \mathtt{AFF}[\vee, \exists]$.

As a direct consequence, we have that KROM and KROM[∃] (resp. AFF and AFF[∃], KROM[∨] and KROM[∨, ∃], AFF[∨] and AFF[∨, ∃]) are both (pairwise) equally succinct and equally expressive.

Accordingly, we focus in the following on the sixteen languages: KROM, HORN, K/H, renH, AFF, HORN[∃], K/H[∃], renH[∃], KROM[∨], HORN[∨], K/H[∨], renH[∨], AFF[∨], HORN[∨, ∃], K/H[∨, ∃], and renH[∨, ∃].

From Proposition 1, we get that:

$$\mathtt{KROM} \subseteq \mathtt{KROM}[\vee] \subseteq \mathtt{KROM}[\vee, \exists]$$

$$\mathtt{HORN} \subseteq \mathtt{HORN}[\exists] \subseteq \mathtt{HORN}[\vee, \exists]$$

$$\mathtt{HORN} \subseteq \mathtt{HORN}[\vee] \subseteq \mathtt{HORN}[\vee, \exists]$$

$$\mathtt{K/H} \subseteq \mathtt{K/H}[\exists] \subseteq \mathtt{K/H}[\vee, \exists]$$

$$\mathtt{K/H} \subseteq \mathtt{K/H}[\vee] \subseteq \mathtt{K/H}[\vee, \exists]$$

$$\mathtt{renH} \subseteq \mathtt{renH}[\exists] \subseteq \mathtt{renH}[\vee, \exists]$$

$$\mathtt{renH} \subseteq \mathtt{renH}[\vee] \subseteq \mathtt{renH}[\vee, \exists]$$

$$\mathtt{AFF} \subseteq \mathtt{AFF}[\vee]$$

Obviously enough, from the definitions of the languages KROM, HORN, K/H, and renH, we also have the following inclusions:

$$\mathtt{KROM} \subseteq \mathtt{K/H}$$

$$\mathtt{HORN} \subseteq \mathtt{K/H}$$

$$\mathtt{HORN} \subseteq \mathtt{renH}$$

Table 1
KROM, HORN, K/H, AFF, renH, their disjunction, existential and full disjunctive closures and the corresponding polynomial-time queries. $\surd$ means "satisfies" and $\circ$ means "does not satisfy unless P = NP".

	CO	VA	CE	IM	EQ	SE	CT	ME	MC
renH[$\vee, \exists$]	$\surd$	$\circ$	$\surd$	$\circ$	$\circ$	$\circ$	$\circ$	$\surd$	$\surd$
K/H[$\vee, \exists$]	$\surd$	$\circ$	$\surd$	$\circ$	$\circ$	$\circ$	$\circ$	$\surd$	$\surd$
HORN[$\vee, \exists$]	$\surd$	$\circ$	$\surd$	$\circ$	$\circ$	$\circ$	$\circ$	$\surd$	$\surd$
AFF[$\vee$]	$\surd$	$\circ$	$\surd$	$\circ$	$\circ$	$\circ$	$\circ$	$\surd$	$\surd$
renH[$\vee$]	$\surd$	$\circ$	$\surd$	$\circ$	$\circ$	$\circ$	$\circ$	$\surd$	$\surd$
K/H[$\vee$]	$\surd$	$\circ$	$\surd$	$\circ$	$\circ$	$\circ$	$\circ$	$\surd$	$\surd$
HORN[$\vee$]	$\surd$	$\circ$	$\surd$	$\circ$	$\circ$	$\circ$	$\circ$	$\surd$	$\surd$
KROM[$\vee$]	$\surd$	$\circ$	$\surd$	$\circ$	$\circ$	$\circ$	$\circ$	$\surd$	$\surd$
renH[$\exists$]	$\surd$	$\surd$	$\surd$	$\surd$	$\circ$	$\circ$	$\circ$	$\surd$	$\surd$
K/H[$\exists$]	$\surd$	$\surd$	$\surd$	$\surd$	$\circ$	$\circ$	$\circ$	$\surd$	$\surd$
HORN[$\exists$]	$\surd$	$\surd$	$\surd$	$\surd$	$\circ$	$\circ$	$\circ$	$\surd$	$\surd$
AFF	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
renH	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\circ$	$\surd$	$\surd$
K/H	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\circ$	$\surd$	$\surd$
HORN	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\circ$	$\surd$	$\surd$
KROM	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\circ$	$\surd$	$\surd$

From such results, we immediately derive that:

$\mathtt{KROM}[\vee] \subseteq \mathtt{K/H}[\vee]$

$\mathtt{HORN}[\vee] \subseteq \mathtt{K/H}[\vee]$

$\mathtt{HORN}[\vee] \subseteq \mathtt{renH}[\vee]$

$\mathtt{HORN}[\exists] \subseteq \mathtt{K/H}[\exists]$

$\mathtt{HORN}[\exists] \subseteq \mathtt{renH}[\exists]$

$\mathtt{HORN}[\vee, \exists] \subseteq \mathtt{K/H}[\vee, \exists]$

$\mathtt{HORN}[\vee, \exists] \subseteq \mathtt{renH}[\vee, \exists]$

In addition, since every consistent KROM formula is a renH formula[5] and since KROM satisfies **CO**, with every K/H formula we can associate in polynomial time an equivalent renH formula, i.e., $\mathtt{K/H} \geq_p \mathtt{renH}$. As a consequence, we also get that

$\mathtt{K/H}[\vee] \geq_p \mathtt{renH}[\vee]$

$\mathtt{K/H}[\exists] \geq_p \mathtt{renH}[\exists]$

$\mathtt{K/H}[\vee, \exists] \geq_p \mathtt{renH}[\vee, \exists]$

Finally, since for every subset $\mathcal{L}$ of C-QDAG, $\mathcal{L} \subseteq \mathcal{L}[\vee] \subseteq \mathcal{L}[\vee, \exists]$, $\subseteq$ is included into $\geq_p$, and both $\subseteq$ and $\geq_p$ are transitive relations, a number of additional inclusions/polynomial translatability results can be directly obtained from the results above; they will be exploited in some forthcoming proofs.

5.1. Queries and transformations

As to queries, we have obtained the following results:

Proposition 5. *The results in Table 1 hold.*

As to transformations, we have obtained the following results:

Proposition 6. *The results in Table 2 hold.*

[5] This is a direct consequence of the fact that a CNF formula α is renamable Horn precisely when there exists an interpretation ω such that at most one literal per clause of α is false in ω [26]; indeed, when α is a KROM formula, this last statement exactly means that α is consistent.

Table 2
KROM, HORN, K/H, AFF, renH, their disjunction, existential and full disjunctive closures and the corresponding polynomial-time transformations. $\surd$ means "satisfies," $\bullet$ means "does not satisfy," while $\circ$ means "does not satisfy unless P = NP." ! means that the transformation is not always feasible within the language.

	CD	FO	SFO	EN	SEN	$\wedge$C	$\wedge$BC	$\vee$C	$\vee$BC	$\neg$C
renH[$\vee$, $\exists$]	$\surd$	$\surd$	$\surd$	$\circ$	$\circ$	$\circ$	$\circ$	$\surd$	$\surd$	$\circ$
K/H[$\vee$, $\exists$]	$\surd$	$\surd$	$\surd$	$\circ$	$\surd$	$\circ$	$\circ$	$\surd$	$\surd$	$\circ$
HORN[$\vee$, $\exists$]	$\surd$	$\surd$	$\surd$	$\circ$	$\surd$	$\circ$	$\surd$	$\surd$	$\surd$	$\circ$
AFF[$\vee$]	$\surd$	$\surd$	$\surd$	$\circ$	$\surd$	$\circ$	$\surd$	$\surd$	$\surd$	$\circ$
renH[$\vee$]	$\surd$	$\bullet$	$\surd$	$\circ$	$\circ$	$\circ$	$\circ$	$\surd$	$\surd$	$\circ$
K/H[$\vee$]	$\surd$	$\bullet$	$\surd$	$\circ$	$\surd$	$\circ$	$\circ$	$\surd$	$\surd$	$\bullet$
HORN[$\vee$]	$\surd$	$\bullet$	$\surd$	$\circ$	$\surd$	$\circ$	$\surd$	$\surd$	$\surd$	$\bullet$
KROM[$\vee$]	$\surd$	$\surd$	$\surd$	$\circ$	$\surd$	$\circ$	$\surd$	$\surd$	$\surd$	$\bullet$
renH[$\exists$]	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	!	!	!	!	!
K/H[$\exists$]	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	!	!	!	!	!
HORN[$\exists$]	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	!	!	!
AFF	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	!	!	!
renH	$\surd$	$\bullet$	$\surd$	$\surd$	$\surd$	!	!	!	!	!
K/H	$\surd$	$\bullet$	$\surd$	$\surd$	$\surd$	!	!	!	!	!
HORN	$\surd$	$\bullet$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	!	!	!
KROM	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	!	!	!

5.2. Expressiveness

It is well-known that none of the languages KROM, HORN, K/H, renH, or AFF is complete for propositional logic. For instance, there is no formula from any of these languages which is equivalent to the CNF formula $(x \vee y \vee z) \wedge (\neg x \vee \neg y \vee \neg z)$. This is problematic for many applications; indeed, what can be done when the available information cannot be represented in the targeted language? Approximating it is not always an option, especially because the best approximation of the available information can be rough and the missing pieces of information in the approximation can be crucial ones for reasoning and/or decision making. In the following, we are going to prove that while considering the existential closure of any of those languages does not increase its expressiveness, switching to its disjunction closure (or to its full disjunctive closure) is enough to recover a complete propositional language, thus escaping from the above mentioned expressiveness problem.

Let us start with the existential closures of KROM, HORN, K/H, renH, and AFF. First, since KROM (resp. AFF) is polynomially equivalent to KROM[$\exists$] (resp. AFF[$\exists$]), it turns out that those languages are (pairwise) equally expressive: KROM[$\exists$] $\sim_e$ KROM, and AFF[$\exists$] $\sim_e$ AFF. Similarly, we have derived the following expressiveness results, showing that the existential closure of any language $\mathcal{L}$ among HORN, K/H, and renH is not more expressive than $\mathcal{L}$ itself.

Proposition 7.

- HORN[$\exists$] $\sim_e$ HORN.
- K/H[$\exists$] $\sim_e$ K/H.
- renH[$\exists$] $\sim_e$ renH.

Now, from the definitions of KROM, HORN, K/H, renH, the fact that K/H $\geq_p$ renH, and the fact that $x \vee y$ is a KROM formula, which is not equivalent to a HORN one, $\neg x \vee \neg y \vee \neg z$ is a HORN formula which is not equivalent to a KROM one, $x \vee y \vee z$ is a renH formula which is not equivalent to a K/H one, we easily get that:

$$\text{KROM} \not\leq_e \text{HORN} \quad \text{and} \quad \text{HORN} \not\leq_e \text{KROM}$$

$$\text{renH} <_e \text{K/H} <_e \text{HORN}$$

$$\text{K/H} <_e \text{KROM}$$

In addition, AFF and any of KROM, HORN, K/H, renH are incomparable w.r.t. $\leq_e$. Indeed, there is no renH formula equivalent to the AFF formula $x \oplus y \oplus z$. This comes from the fact that every CNF formula equivalent to $x \oplus y \oplus z$ must contain the four clauses $x \vee y \vee z$, $\neg x \vee \neg y \vee z$, $x \vee \neg y \vee \neg z$, $\neg x \vee y \vee \neg z$ since those clauses are essential prime implicates of $x \oplus y \oplus z$, plus the fact that by construction, every CNF formula containing the clauses $x \vee y \vee z$, $\neg x \vee \neg y \vee z$, $x \vee \neg y \vee \neg z$, $\neg x \vee y \vee \neg z$ is not renamable Horn (renaming at least two variables in the first clause to make it a Horn clause also changes one of the remaining three clauses into a non-Horn one). Conversely, there is no AFF formula equivalent to $\neg x \vee \neg y$, which is both in KROM and in HORN. This is a direct consequence of the semantical characterization result concerning AFF recalled in Section 2.3: with $x < y$, $\omega_1 = 00$, $\omega_2 = 01$, and $\omega_3 = 10$ are models of $\neg x \vee \neg y$, but $\oplus(\omega_1, \omega_2, \omega_3) = 11$ is not a model of $\neg x \vee \neg y$.

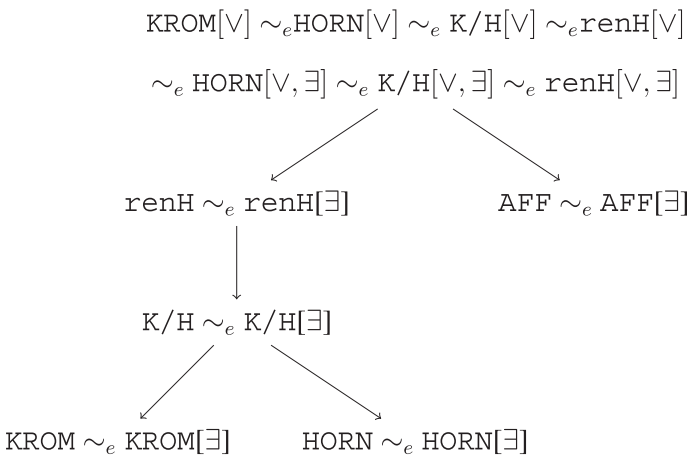

Fig. 5. The expressiveness picture for disjunctive closures. An arrow from $\mathcal{L}_1$ to $\mathcal{L}_2$ means that $\mathcal{L}_1$ is strictly more expressive than $\mathcal{L}_2$, so that a lack of arrow means that the expressiveness of $\mathcal{L}_1$ and the expressiveness of $\mathcal{L}_2$ are incomparable.

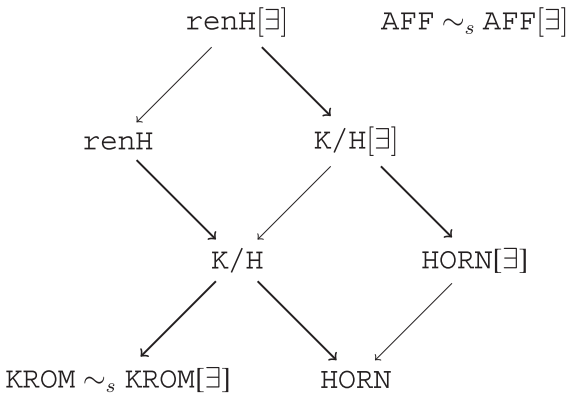

Fig. 6. The succinctness picture for incomplete languages. An arrow from $\mathcal{L}_1$ to $\mathcal{L}_2$ means that $\mathcal{L}_1$ is strictly more succinct than $\mathcal{L}_2$, i.e., $\mathcal{L}_1 <_s \mathcal{L}_2$. The arrow is thick in the specific case when the fact that $\mathcal{L}_2 \not\leq_s \mathcal{L}_1$ comes from the fact that $\mathcal{L}_2 \not\leq_e \mathcal{L}_1$. A lack of arrow means that the succinctness of $\mathcal{L}_1$ and the succinctness of $\mathcal{L}_2$ are incomparable.

Let us finally switch to the disjunction closures and the full disjunctive closures of KROM, HORN, K/H, renH, or AFF; interestingly, the eight languages defined as such are equally, and fully, expressive:

Proposition 8. KROM[∨], HORN[∨], K/H[∨], renH[∨], AFF[∨], HORN[∨, ∃], K/H[∨, ∃], renH[∨, ∃] *are complete propositional languages.*

Fig. 5 depicts the expressiveness relationships identified in the above propositions.

5.3. Succinctness

As to incomplete languages, since KROM (resp. AFF) is polynomially equivalent to KROM[∃] (resp. AFF[∃]), those languages are (pairwise) equally succinct: KROM[∃] $\sim_s$ KROM, and AFF[∃] $\sim_s$ AFF. More interestingly, we have obtained the following succinctness results, showing that the existential closure of any language $\mathcal{L}$ among HORN, K/H, and renH is strictly more succinct than $\mathcal{L}$ itself.

Proposition 9.

- HORN[∃] $<_s$ HORN.
- K/H[∃] $<_s$ K/H.
- renH[∃] $<_s$ renH.
- renH *and* K/H[∃] *are incomparable w.r.t.* $\leq_s$.
- K/H *and* HORN[∃] *are incomparable w.r.t.* $\leq_s$.

Fig. 6 summarizes the succinctness relationships among incomplete languages identified in Proposition 9. We observe that it does not coincide with the corresponding expressiveness picture, restricted to incomplete languages (see Fig. 5).

Table 3
The relative succinctness of the full disjunctive closures of KROM, HORN, AFF, K/H, and renH.

	AFF[∨]	renH[∨, ∃]	K/H[∨, ∃]	HORN[∨, ∃]	KROM[∨]
AFF[∨]	$\sim_s$	$\not\leq_s$	$\not\leq_s$	$\not\leq_s$	$\not\leq_s$
renH[∨, ∃]	$\not\leq_s$	$\sim_s$	$\leq_s$	$\leq_s$	$\leq_s$
K/H[∨, ∃]	$\not\leq_s$	$\not\leq_s$	$\sim_s$	$\leq_s$	$\leq_s$
HORN[∨, ∃]	$\not\leq_s$	$\not\leq_s$	$\not\leq_s$	$\sim_s$	$\not\leq_s$
KROM[∨]	$\not\leq_s$	$\not\leq_s$	$\not\leq_s$	$\not\leq_s$	$\sim_s$

As to complete languages, our succinctness results mainly focus on the five languages KROM[∨], HORN[∨, ∃], K/H[∨, ∃], renH[∨, ∃], AFF[∨], the full disjunctive closures of the incomplete languages KROM, HORN, K/H, renH, AFF considered at start.[6]

There are several reasons for this focus:

- KROM[∨], HORN[∨, ∃], K/H[∨, ∃], renH[∨, ∃], AFF[∨] are complete languages, while KROM, HORN, K/H, renH, AFF and their existential closures are not (see Proposition 7 and Proposition 8 above).
- HORN[∨, ∃], K/H[∨, ∃], renH[∨, ∃] satisfy the same queries as the corresponding disjunction closures, namely HORN[∨], K/H[∨], renH[∨] (see Proposition 5), and more transformations than them (see Proposition 6), since they offer **FO** "for free".
- due to the obvious inclusion HORN[∨] ⊆ HORN[∨, ∃], we have that HORN[∨, ∃] is at least as succinct as HORN[∨]; and similarly for K/H[∨, ∃] and renH[∨, ∃].

Actually, we can strengthen this point by proving that the full disjunctive closure of HORN (resp. K/H, renH) is strictly more succinct than the corresponding disjunction closure:

Proposition 10.

- HORN[∨, ∃] $<_s$ HORN[∨].
- K/H[∨, ∃] $<_s$ K/H[∨].
- renH[∨, ∃] $<_s$ renH[∨].

Thus, the full disjunctive closures of the incomplete languages KROM, HORN, K/H, renH, AFF are either equally succinct as the corresponding disjunction closures (this is the case for the closures of KROM and of AFF), or strictly more succinct than them (for the three remaining languages).

Let us now provide the remaining succinctness results we got. We split our results into two propositions (and two tables). In the first table, we compare KROM[∨], HORN[∨, ∃], K/H[∨, ∃], renH[∨, ∃], AFF[∨] w.r.t. spatial efficiency $\leq_s$.

Proposition 11. *The results in Table 3 hold.*

As a direct consequence of Proposition 11, we have that

$$\text{renH}[\vee, \exists] <_s \text{K/H}[\vee, \exists] <_s \text{HORN}[\vee, \exists]$$

$$\text{K/H}[\vee, \exists] <_s \text{KROM}[\vee, \exists]$$

One can observe that the resulting succinctness picture is similar to the expressiveness picture for the corresponding incomplete fragments AFF, renH, K/H, HORN, KROM.

In Proposition 12, we compare w.r.t. $\leq_s$ the languages KROM[∨], HORN[∨, ∃], K/H[∨, ∃], renH[∨, ∃] and AFF[∨] with several classes of propositional representations for KC which have been introduced so far, and with CNF. We specifically focus on those target classes for which compilers have been developed, i.e., PI, IP, DNF, $\text{OBDD}_<$, d-DNNF, and DNNF_T.

Proposition 12. *The results in Table 4 hold.*

Fig. 7 depicts the succinctness relationships reported mainly in Proposition 11 and Proposition 12. The closure languages considered in this paper are underlined.

[6] Remember that KROM[∨, ∃] $\sim_p$ KROM[∨] and that AFF[∨, ∃] $\sim_p$ AFF[∨], see Proposition 4.

Table 4
Comparing w.r.t. succinctness the full disjunctive closures of KROM, HORN, K/H, renH, and AFF, with $\text{OBDD}_<$, IP, DNF, d-DNNF, DNNF_T, PI, and CNF. * means that the result holds unless the polynomial hierarchy collapses.

	AFF[∨]	renH[∨, ∃]	K/H[∨, ∃]	HORN[∨, ∃]	KROM[∨, ∃]
CNF	$\not\leq_s, \not\geq_s$	$\not\leq_s, \not\geq_s$	$\not\leq_s, \not\geq_s$	$\not\leq_s, \not\geq_s$	$\not\leq_s, \not\geq_s$
PI	$\not\leq_s, \not\geq_s$	$\not\leq_s, \not\geq_s$	$\not\leq_s, \not\geq_s$	$\not\leq_s, \not\geq_s$	$\not\leq_s, \not\geq_s$
DNNF_T	$\not\leq_s, \not\geq_s$	$\not\leq_s, \not\geq_s$	$\not\leq_s, \not\geq_s$	$\not\leq_s, \not\geq_s$	$\not\leq_s, \not\geq_s$
d-DNNF	$\not\leq_s^*, \not\geq_s$	$\not\leq_s^*, \not\geq_s$	$\not\leq_s^*, \not\geq_s$	$\not\leq_s^*, \not\geq_s$	$\not\leq_s^*, \not\geq_s$
DNF	$\not\leq_s, \geq_s$	$\not\leq_s, \geq_s$	$\not\leq_s, \geq_s$	$\not\leq_s, \geq_s$	$\not\leq_s, \geq_s$
IP	$\not\leq_s, \geq_s$	$\not\leq_s, \geq_s$	$\not\leq_s, \geq_s$	$\not\leq_s, \geq_s$	$\not\leq_s, \geq_s$
$\text{OBDD}_<$	$\not\leq_s, \not\geq_s$	$\not\leq_s, \not\geq_s$	$\not\leq_s, \not\geq_s$	$\not\leq_s, \not\geq_s$	$\not\leq_s, \not\geq_s$

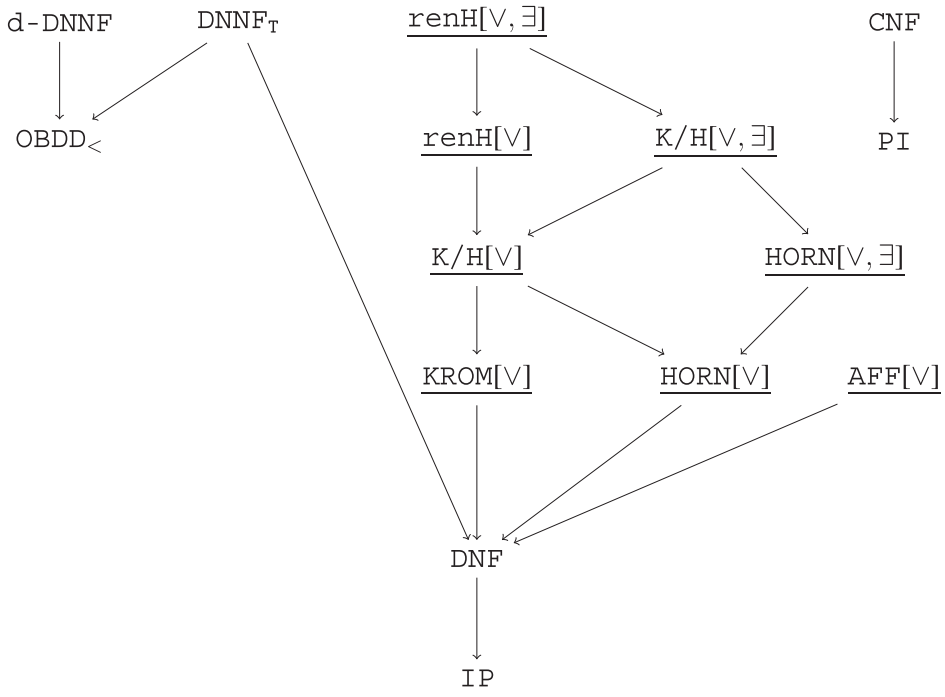

Fig. 7. The succinctness picture for complete classes. An arrow from $\mathcal{L}_1$ to $\mathcal{L}_2$ means that $\mathcal{L}_1$ is strictly more succinct than $\mathcal{L}_2$, so that a lack of arrow means that the succinctness of $\mathcal{L}_1$ and the succinctness of $\mathcal{L}_2$ are incomparable for sure (or, under the assumption that the polynomial hierarchy does not collapse when d-DNNF is concerned). For a clarity sake, there are two exceptions to this notation, which also concern d-DNNF: it is unknown whether d-DNNF $\leq_s$ PI, and whether d-DNNF $\leq_s$ IP [1].

5.4. Discussion

Let us now compare in more details the five languages KROM, HORN, K/H, renH, and AFF, with their closures and with other classes of propositional representations considered so far for knowledge compilation.

We start with the existential closures. Since applying the existential closure principle to any of KROM, HORN, K/H, renH, and AFF does not change its expressiveness, the existential closures of KROM, HORN, K/H, renH, and AFF are incomplete languages as well. KROM[∃] (resp. AFF[∃]) is polynomially equivalent to KROM (resp. AFF) since KROM and AFF satisfy **FO**. As a consequence, KROM[∃] and AFF[∃] satisfy the same queries and transformations as their underlying language, and KROM[∃] and AFF[∃] are equally succinct as KROM and AFF, respectively. For the remaining existential closures (namely, HORN[∃], K/H[∃], renH[∃]), all the transformations already offered by HORN, K/H, renH, are preserved and **FO** is obtained "for free". However, some queries offered by HORN, K/H, and renH (**EQ**, **SE**) are not preserved. This seems to be the price to be paid for the gain in succinctness the existential closures offer. Indeed, we have HORN[∃] $<_s$ HORN, K/H[∃] $<_s$ K/H, and renH[∃] $<_s$ renH. Thus, for applications where their expressiveness proves enough and **EQ** and **SE** are not expected but **FO** is, renH[∃] (resp. HORN[∃]) appears as a better choice than renH (resp. HORN) as a target language for KC.

Unlike existential closures, the disjunction closures and the full disjunctive closures of KROM, HORN, K/H, renH, and AFF are complete propositional languages, i.e., fully expressive ones. Furthermore, switching from any of KROM, HORN, K/H, renH, or AFF to its disjunction closure or its full disjunctive closure leads to get ∨**C** (hence ∨**BC**) "for free" and **FO**, when it was not already offered. Conversely, some queries and transformations primarily offered are then lost; as to queries, this is the case for **VA**, **IM**, **EQ**, **SE** for the five languages, plus **CT** satisfied by AFF but not by any of its disjunction closure or its full disjunctive closure; as to transformations, this is the case for **EN** and ∧**C** (and even **SEN**, which is satisfied by renH but not satisfied by renH[∨] or renH[∨, ∃], unless P = NP). Just like considering the existential closures of HORN, K/H, renH leads to strictly more succinct languages, considering the existential closures of HORN[∨], K/H[∨], renH[∨] leads as

well to strictly more succinct languages since HORN[$\vee, \exists$] $<_s$ HORN[$\vee$], K/H[$\vee, \exists$] $<_s$ K/H[$\vee$], and renH[$\vee, \exists$] $<_s$ renH[$\vee$]. Thus, it turns out that *the full disjunctive closures of* KROM*,* HORN*,* K/H*,* renH*, and* AFF *are always at least as interesting as the corresponding disjunction closures from a KC perspective*: as to KROM and AFF, those closures are polynomially equivalent, hence equally interesting; as to HORN, K/H, renH, both closures satisfy the same queries, while each full disjunctive closure offers **FO** (not satisfied by the corresponding disjunction closure) and is strictly more succinct than the underlying language.

Comparing now one another the full disjunctive closures of KROM, HORN, K/H, renH, AFF it turns out that *none of them is strictly dominated* by another one from the KC point of view. All of them are equally expressive, and they satisfy precisely the same queries **CO**, **CE**, **ME**, **MC**. As to transformations, KROM[$\vee$], HORN[$\vee, \exists$], and AFF[$\vee$] satisfy **CD**, **FO**, **SFO**, **SEN**, $\wedge$**BC**, $\vee$**C** and $\vee$**BC**. They are pairwise incomparable w.r.t. succinctness. While K/H[$\vee, \exists$] is strictly more succinct than each of KROM[$\vee$], or HORN[$\vee, \exists$], it does not offer **SEN**, and while renH[$\vee, \exists$] is strictly more succinct than K/H[$\vee, \exists$], it does not offer $\wedge$**BC**.

Finally, it is interesting to compare the full disjunctive closures of KROM, HORN, K/H, renH, AFF, with previous complete classes of representations for propositional logic, which have been considered as target classes for KC. One focuses on IP, DNF, PI, $\text{OBDD}_<$, DNNF_T, and d-DNNF:

- IP satisfies all the queries but **CT**, and no transformation but **CD**, **EN**, **SEN** and $\wedge$**BC**. Hence each of the full disjunctive closures of KROM, HORN, AFF satisfies less queries than IP but they are incomparable w.r.t. transformations. Furthermore, IP is strictly less succinct than any of the full disjunctive closures of KROM, HORN, K/H, renH, AFF.
- DNF satisfies the same queries as any of the full disjunctive closures of KROM, HORN, K/H, renH, AFF, and the same transformations as KROM[$\vee$], HORN[$\vee, \exists$], and AFF[$\vee$]. Since it is strictly less succinct than any of the full disjunctive closures of KROM, HORN, K/H, renH, AFF, it appears as dominated by KROM[$\vee$], HORN[$\vee, \exists$], and AFF[$\vee$].
- PI satisfies all the queries but **CT**, and the transformations **CD**, **FO**, **SFO**, $\vee$**BC**. Hence, any of the full disjunctive closures of KROM, HORN, K/H, renH, AFF satisfies more transformations than PI. In addition, PI is incomparable w.r.t. succinctness with any of them.
- $\text{OBDD}_<$ satisfies all the queries and the transformations **CD**, **SFO**, **SEN**, $\wedge$**BC**, $\vee$**BC**, and $\neg$**C**. Hence it offers more queries than the full disjunctive closures of KROM, HORN, K/H, renH, AFF, but it is incomparable with any of them when transformations are considered. $\text{OBDD}_<$ is also incomparable w.r.t. succinctness with any of the full disjunctive closures of KROM, HORN, K/H, renH, AFF.
- DNNF_T satisfies the same queries as any of the full disjunctive closures of KROM, HORN, K/H, renH, AFF, and the same transformations as KROM[$\vee$], HORN[$\vee, \exists$], and AFF[$\vee$]. It is incomparable w.r.t. succinctness with any of them.
- d-DNNF satisfies all the queries but **SE**, and it is unknown whether it offers **EQ**. As to transformations, it offers only **CD** (it is unknown whether it satisfies **EN**, **SEN**, or $\neg$**C**, but it is known that it does not satisfy the other transformations). Thus, the full disjunctive closures of KROM, HORN, K/H, renH, AFF satisfy less queries than d-DNNF but offer additional transformations. Furthermore, d-DNNF is incomparable w.r.t. succinctness with any of the full disjunctive closures of KROM, HORN, K/H, renH, AFF (unless the polynomial hierarchy collapses).

Thus, *none of the full disjunctive closures of* KROM*,* HORN*,* K/H*,* renH*, and* AFF *is strictly dominated* by any of IP, DNF, PI, $\text{OBDD}_<$, DNNF_T, and d-DNNF, viewing the set of queries, the set of transformations and the succinctness relation as comparison criteria.

6. Conclusion and perspectives

6.1. Conclusion

In the light of the results reported in the previous sections, the following conclusions can be drawn.

Generally speaking, the disjunctive closures of classes $\mathcal{L}$ of propositional representations appear as interesting target classes for KC when the application under consideration expects tractability for the queries and transformations **CO**, **CD** and their consequences (e.g., **CE**, **ME**), as well as **FO** and/or $\vee$**C** (depending on the type of closure which is considered). Especially, as soon as $\mathcal{L}$ is stable by uniform renaming, the transformations **FO**, $\vee$**C** are offered "for free" by the full disjunctive closure $\mathcal{L}[\vee, \exists]$ (even if the underlying class $\mathcal{L}$ does not offer any of them), while **CO**, **CD** are preserved by the closure. The other queries and transformations considered in the KC map are not guaranteed to be offered or to survive a disjunctive closure operation in the general case.

Considering specific disjunctive closures may allow for preserving additional queries or transformations, and for increasing the expressiveness of the underlying language. Thus, the disjunction closure and the full disjunctive closure of any language $\mathcal{L}$ containing TERM are complete propositional languages, even if $\mathcal{L}$ is not (KROM, HORN, K/H, renH, and AFF are such languages). Clearly enough, fully expressive propositional languages are highly expected by many applications.

Of course, it cannot be guaranteed in the general case that the size of a compiled form remains "small enough" when a disjunctive closure is targeted. Nevertheless, every disjunctive closure of a class $\mathcal{L}$ includes $\mathcal{L}$ as a subset, hence applying a disjunctive closure principle to a class $\mathcal{L}$ decreases neither the expressiveness nor the succinctness of $\mathcal{L}$. Actually, applying any/both of those two principles may lead to new classes, which can prove strictly more expressive and strictly more succinct than the underlying class $\mathcal{L}$. Thus, each of the disjunction closure and the full disjunctive closure of any of KROM, HORN, K/H, renH, and AFF is strictly more expressive than the underlying language. Furthermore, the full disjunctive clo-

sure HORN[$\vee, \exists$] (resp. K/H[$\vee, \exists$], renH[$\vee, \exists$]) is strictly more succinct than the corresponding disjunction closure HORN[$\vee$] (resp. K/H[$\vee$], renH[$\vee$]).

Now, from the application point of view, there are many important problems in AI and in other fields of Computer Science, where one is interested in encoding some pieces of information using representations for which **CO**, **CD**, **FO** and **ME** are computationally easy.

For instance, in model-based diagnosis, it makes sense to compile the description of the system to be diagnosed (during an off-line phase) in order to be able to generate efficiently consistency-based diagnoses, for a number of observations available on-line only [33,41,42]. Such diagnoses are the models of the system description, once conditioned by the given observation and then projected onto the variables expressing the components statuses (in the simplest case, faulty or not). Accordingly, if the system description has been compiled first into a representation which satisfies **CO**, **CD**, **FO** and **ME**, then the diagnoses can be computed in input-output polynomial time. Our results thus show full disjunctive closures of languages $\mathcal{L}$ satisfying the stability by uniform renaming condition as valuable target languages for the compilation, as soon as $\mathcal{L}$ satisfies **CO** and **CD** (which is the case for KROM, HORN, K/H, renH, and AFF).

In product configuration and interactive recommendation, it is also important to offer some response-time guarantees to the front-end user, especially when the interaction is Web-based. In order to achieve this goal, an approach consists in compiling the product catalog into a propositional representation (the models of it representing the feasible products). Among the operations required by the configuration process are propagating the user's choices (the **CD** transformation), testing whether at least one feasible product is compatible with the user's choices (the **CO** query), and listing a fixed number of feasible products compatible with the user's choices (see e.g. [43,44]). Often, the feasible products are described using two types of variables (or "codes" [45]): the customer variables – the variables the user controls – and the manufacturer control variables – which express some information related to the factory or to the distribution of the product, and are not available to the user. Thus, the manufacturer control variables must be forgotten from the representation before listing the solutions. Our results show that those operations can be achieved efficiently when the catalog has been compiled into a full disjunctive closure of a class $\mathcal{L}$ of propositional representations, stable by uniform renaming, and satisfying **CO** and **CD**. In particular, the task of enumerating a preset number of solutions is feasible in polynomial time in this case (Algorithm 1 given in Appendix A is a polynomial delay enumeration procedure).

Beyond AI applications, enumerating models once projected on a given set of variables appears as a fundamental issue for a number of problems considered in software engineering and formal methods. Thus, in the setting of automatic case generation based on propositional logic, such models correspond to test cases [46]. The problem ALL-SAT (or "all-solutions" SAT) which consists in enumerating the assignments to "important" variables of a propositional representation, which can be extended to models, turns out to be very significant in symbolic model checking [47], which explains that dedicated algorithms have been developed for solving it [48]. Indeed, this problem is considered for predicate abstraction [49], and re-parameterization in symbolic simulation [50]. In reachability analysis, one is interested in computing the set of states reachable from (resp. leading to) a given set of states under a transition relation; this is called the image (resp. pre-image) computation problem. The transition relation T can be modeled as a Boolean function T over $X \cup Y \cup X'$, complete terms γ_X over X (resp. X') are used to denote states before (resp. after) a transition and complete terms γ_Y over Y represent inputs making precise the transition. By construction, the models of $\exists Y.T_{|\gamma_X}$ (resp. $\exists Y.T_{|\gamma_{X'}}$) represent the image of γ_X (resp. the pre-image of $\gamma_{X'}$) by T. The "important" variables are those of X' (resp. X). Accordingly, many SAT solvers have been customized into ALL-SAT solvers precisely for computing images or pre-images (see e.g. [51,52]) from CNF representations of transition relations. In practice, such SAT-based approaches to symbolic model checking can prove much more efficient than OBDD$_<$-based approaches on some instances, which coheres with the fact that the succinctness of OBDD$_<$ and the succinctness of CNF are incomparable [53]. Interestingly, when T is represented as a full disjunctive closure of a class $\mathcal{L}$ of propositional representations, stable by uniform renaming, and satisfying **CO** and **CD**, both the computation of $\exists Y.T_{|\gamma_X}$ (resp. $\exists Y.T_{|\gamma_{X'}}$) and the enumeration of its models can be achieved in polynomial time (in the size of the input plus the size of the output). Contrastingly, no response-time guarantee can be ensured in the general case for computing a single model when T is represented as a CNF formula.

Thus, for each of the applications above, considering full disjunctive closures for the representation purpose can prove to be a reasonable choice.

6.2. Perspectives

This work calls for several perspectives.

One of them concerns the problem of closed-world reasoning. Indeed, the disjunction covers of HORN and renH are known as interesting target languages when propositional formulae are to be interpreted under some form of the closed-world assumption, like the extended closed-world assumption (ECWA) [54], the extended generalized closed-world assumption (EGCWA) [55], the generalized closed-world assumption (GCWA) [56] or the careful closed-world assumption (CCWA) [57]. To be more precise, though inference from a propositional formula interpreted under ECWA, EGCWA, GCWA or CCWA is Π_2^p-hard, its complexity is at most at the first level of the polynomial hierarchy when the formula belongs to HORN[$\vee$] or to renH[$\vee$] [58]. Furthermore, the complexity of inference under EGCWA falls down to P when HORN[$\vee$] formulae are considered, or when GCWA is considered and queries are limited to CNF formulae. Finally, it turns out that the complexity of closed-world reasoning is the same one for HORN[$\vee$] formulae and for DNF formulae, despite the fact

that DNF is strictly less succinct than HORN[∨]. It would be interesting to identify the complexity of closed-world reasoning for full disjunctive closures, especially those of HORN and renH.

Another important issue for further research is the design and the evaluation of compilers targeting the disjunctive closures introduced in the paper. Actually, compilers targeting some of those closures considered here do exist. Thus, Boufkhad et al. [6] present some compilation algorithms targeting KROM[∨], HORN[∨], K/H[∨], and renH[∨], and evaluate them on a number of benchmarks. While the obtained results show the feasibility of computing disjunction closure compilations, we can hardly use them to compare the practical significance of the corresponding closures with $\mathtt{OBDD}_<$ and $\mathtt{DNNF}_\mathrm{T}$ for which some experimental results are also available. Indeed, the compilation algorithms given in [6] are based on an old-style DPLL SAT solver, and the performances of such solvers are dramatically overtaken by those of modern SAT solvers, based on a CDCL architecture.

Interestingly, Nishimura et al. [59] have shown that the problem of determining whether a given CNF formula α has a strong KROM-backdoor set (resp. a strong HORN-backdoor set) containing at most k variables is fixed-parameter tractable with parameter k. Similarly, Samer and Szeider [60] have shown that the problem of determining whether a given CNF formula α has a KROM-backdoor tree (resp. a HORN-backdoor tree) containing at most k leaves is fixed-parameter tractable with parameter k. The algorithm given in [60] can be used to determine "efficiently" (i.e., for sufficiently "small" k) whether a KROM[∨] compilation or a HORN[∨] compilation of "reasonable" size (i.e., linear in k and the size of α) exists. As mentioned in [60]: *"There is some empirical evidence that real-world instances actually have small backdoor sets"*. Such instances also have "small" HORN[∨] representations (hence, "small" K/H[∨] and "small" renH[∨] representations).[7] This explains why it makes sense to develop new compilation algorithms targeting disjunction closures.

Incorporating existentially quantified variables in the representations during the compilation phase in order to generate full disjunctive closures also appears as an interesting perspective. Indeed, new variables can be introduced as "names" given to arbitrarily complex subformulae of the input formula (using equivalences); the point is that equivalence w.r.t. the input formula is preserved when such variables are existentially quantified. Taking advantage of it can dramatically reduce the size of the compiled forms (our succinctness results show that exponential gaps in the representation size can be achieved thanks to existential closure); the difficulty is to determine when introducing new variables (this is reminiscent to the general problem of lemmatization in automated reasoning).

Finally, the fact that each of KROM[∨], HORN[∨, ∃], and AFF[∨] satisfies ∧**BC** paves the way for bottom-up compilation algorithms for those classes. As noted in [18], this is important for applications from formal verification based on unbounded model checking which require bottom-up, incremental compilation of formulae, where pieces of information are compiled independently and then conjoined together. This explains why HORN[∨, ∃], and AFF[∨] which offers **CD**, **FO**, ∧**BC**, and **ME** appear as valuable candidates for the image/pre-image computation problem considered in reachability analysis, as discussed above. Indeed, $\mathtt{OBDD}_<$, which offers **CD**, **FO**, ∧**BC**, and **ME** as well, has been extensively used for the purpose of symbolic model checking [62]; furthermore, we have shown that the succinctness of HORN[∨, ∃], and of AFF[∨] are incomparable with the succinctness of $\mathtt{OBDD}_<$. Since each of KROM[∨], HORN[∨, ∃], and AFF[∨] satisfies **CO** and includes CLAUSE, getting ∧**BC** is optimal in the sense that no class of propositional representations containing CLAUSE can satisfy both ∧**C** and **CO**, unless P = NP.

Acknowledgements

We would like to thank the anonymous reviewers for many helpful comments.

Appendix A. Proofs

Proposition 1. *For every subset $\mathcal{L}$, $\mathcal{L}'$ of C-*QDAG *and every subset $\triangle_1$, $\triangle_2$ of $C \cup \{\exists, \forall\}$, we have:*

0. $\mathcal{L} \subseteq \mathcal{L}[\triangle_1]$, *and if* $\mathcal{L} \subseteq \mathcal{L}'$, *then* $\mathcal{L}[\triangle_1] \subseteq \mathcal{L}'[\triangle_1]$.
1. $(\mathcal{L}[\triangle_1])[\triangle_2] \subseteq \mathcal{L}[\triangle_1 \cup \triangle_2]$.
2. $(\mathcal{L}[\triangle_1])[\triangle_1] = \mathcal{L}[\triangle_1]$.
3. *If* $\triangle_1 \subseteq \triangle_2$ *then* $\mathcal{L}[\triangle_1] \subseteq \mathcal{L}[\triangle_2]$.
4. *If* $\triangle_1 \subseteq \triangle_2$ *then* $(\mathcal{L}[\triangle_1])[\triangle_2] = \mathcal{L}[\triangle_2]$ *and* $(\mathcal{L}[\triangle_2])[\triangle_1] = \mathcal{L}[\triangle_2]$.

Proof.

0. Obvious.
1. $(\mathcal{L}[\triangle_1])[\triangle_2] \subseteq \mathcal{L}[\triangle_1 \cup \triangle_2]$ is immediate from Definition 9; indeed, the construction of any representation from $(\mathcal{L}[\triangle_1])[\triangle_2]$ requires only representations from $\mathcal{L}[\triangle_1]$ and operators in $\triangle_2$; and the construction of any representation from $\mathcal{L}[\triangle_1]$ requires only representations from $\mathcal{L}$ and operators in $\triangle_1$. Thus, if representations from $\mathcal{L}$ and operators in $\triangle_1 \cup \triangle_2$ are available, then any representation from $(\mathcal{L}[\triangle_1])[\triangle_2]$ can be generated.

[7] Note by the way that determining whether "small" renH[∨] representations which are not HORN[∨] representations exist can be computationally demanding since the detection of a strong renH-backdoor set is W[2]-hard [61].

2. $(\mathcal{L}[\triangle_1])[\triangle_1] = \mathcal{L}[\triangle_1]$: considering the inclusion reported at item 1. in this proof with $\triangle_2 = \triangle_1$, we get $(\mathcal{L}[\triangle_1])[\triangle_1] \subseteq \mathcal{L}[\triangle_1]$; the converse inclusion $\mathcal{L}[\triangle_1] \subseteq (\mathcal{L}[\triangle_1])[\triangle_1]$ follows from the inclusion at item 0. in this proof.
3. If $\triangle_1 \subseteq \triangle_2$ then $\mathcal{L}[\triangle_1] \subseteq \mathcal{L}[\triangle_2]$: the inclusion at item 0. in this proof shows that $\mathcal{L}[\triangle_1] \subseteq (\mathcal{L}[\triangle_1])[\triangle_2]$, and item 1. in this proof shows that $(\mathcal{L}[\triangle_1])[\triangle_2] \subseteq \mathcal{L}[\triangle_1 \cup \triangle_2]$. The fact that $\mathcal{L}[\triangle_1 \cup \triangle_2]$ is equal to $\mathcal{L}[\triangle_2]$ when $\triangle_1 \subseteq \triangle_2$ completes the proof.
4. Suppose that $\triangle_1 \subseteq \triangle_2$. Then, from the equality reported at item 2. in this proof, since $\triangle_1 \cup \triangle_2 = \triangle_2$, we have $(\mathcal{L}[\triangle_1])[\triangle_2] \subseteq \mathcal{L}[\triangle_2]$ and $(\mathcal{L}[\triangle_2])[\triangle_1] \subseteq \mathcal{L}[\triangle_2]$. Conversely, the inclusion at item 0. in this proof shows that $\mathcal{L}[\triangle_2] \subseteq (\mathcal{L}[\triangle_2])[\triangle_1]$; finally, $\mathcal{L}[\triangle_2] \subseteq (\mathcal{L}[\triangle_1])[\triangle_2]$ derives from the fact that $\mathcal{L} \subseteq \mathcal{L}[\triangle_1]$ (which is again ensured the inclusion at item 0. in this proof), and the implication reported at item 0. as well. □

Proposition 2. *Let $\mathcal{L}$ be any subset of C-*QDAG *s.t. $\mathcal{L}$ is stable by uniform renaming. We have:*

- $(\mathcal{L}[\exists])[\vee] \sim_p (\mathcal{L}[\vee])[\exists] \sim_p \mathcal{L}[\vee, \exists]$.
- $(\mathcal{L}[\forall])[\wedge] \sim_p (\mathcal{L}[\wedge])[\forall] \sim_p \mathcal{L}[\wedge, \forall]$.

Proof. We just prove the first point of the proposition; the second one is similar (by duality). The facts that $\mathcal{L}[\vee, \exists] \leq_p (\mathcal{L}[\vee])[\exists]$ and $\mathcal{L}[\vee, \exists] \leq_p (\mathcal{L}[\exists])[\vee]$ come immediately from the inclusions $\mathcal{L}[\vee, \exists] \supseteq (\mathcal{L}[\vee])[\exists]$ and $\mathcal{L}[\vee, \exists] \supseteq (\mathcal{L}[\exists])[\vee]$ (see item 1. in Proposition 1). It remains to show that $\mathcal{L}[\vee, \exists] \geq_p (\mathcal{L}[\vee])[\exists]$ and $\mathcal{L}[\vee, \exists] \geq_p (\mathcal{L}[\exists])[\vee]$.

- $\mathcal{L}[\vee, \exists] \geq_p (\mathcal{L}[\vee])[\exists]$. This comes from the possibility to turn in polynomial time any $\mathcal{L}[\vee, \exists]$ representation into an equivalent, prenex, one while preserving the set of free variables. The proof is by structural induction. Let α be any representation from $\mathcal{L}[\vee, \exists]$:
  - If α is an $\mathcal{L}$ representation, then it is also an $(\mathcal{L}[\vee])[\exists]$ representation due to the inclusion $(\mathcal{L}[\vee])[\exists] \supseteq \mathcal{L}$ which comes from item 0. in Proposition 1.
  - If α is not an $\mathcal{L}$ representation and $\alpha = \vee(\beta_1, \ldots, \beta_n)$ with $\beta_i \in \mathcal{L}[\vee, \exists]$ for $i \in 1, \ldots, n$, then by induction hypothesis,[8] one can compute in polynomial time n representations $\beta'_1, \ldots, \beta'_n \in (\mathcal{L}[\vee])[\exists]$ such that for $i \in 1, \ldots, n$, we have $\beta'_i \equiv \beta_i$. Hence, for $i \in 1, \ldots, n$, we can compute in polynomial time $X_i \subseteq PS$ and $\beta''_i \in \mathcal{L}[\vee]$ such that $\beta'_i \equiv \exists X_i.\beta''_i$ (especially, if β'_i is an $\mathcal{L}[\vee]$ representation, then we take $X_i = \emptyset$). For $i \in 1, \ldots, n$, let X'_i be a set of variables of PS which is disjoint with the set of variables occurring in α and such that there exists a bijection between X'_i and X_i. One can always find such a bijection since each β_i $(i \in 1, \ldots, n)$ belongs to $\mathcal{L}[\vee, \exists]$, which is stable by uniform renaming since $\mathcal{L}$ is so. Furthermore, since PS is countably infinite, we can always find sets X'_i so that for $i, j \in 1, \ldots, n$, if $i \neq j$ then $X'_i \cap X'j = \emptyset$. Now, for $i \in 1, \ldots, n$, let $\beta''_i[X_i \leftarrow X'_i]$ be the representation obtained by replacing in a uniform way in β''_i every occurrence of $x \in X_i$ by the corresponding variable $x' \in X'_i$. Clearly enough, such representations can be computed in polynomial time. Since quantified variables are dummy ones, we have $\exists X_i.\beta''_i \equiv \exists X'_i.\beta''_i[X_i \leftarrow X'_i]$. Hence, we have $\alpha \equiv \vee(\beta'_1, \ldots, \beta'_n) \equiv \vee(\exists X_1.\beta''_1, \ldots, \exists X_n.\beta''_n) \equiv \vee(\exists X'_1.\beta''_1[X_1 \leftarrow X'_1], \ldots, \exists X'_n.\beta''_n[X_n \leftarrow X'_n])$. Since for each $i \in 1, \ldots, n$, we have $Var(\beta''_i[X_i \leftarrow X'_i]) \cap \bigcup_{j=1,\ldots,n | j \neq i} X'_j = \emptyset$, each $\exists X'_i.\beta''_i[X_i \leftarrow X'_i]$ is equivalent to $\exists \bigcup_{j=1}^n X'_j.\beta''_i[X_i \leftarrow X'_i]$. Thus we get that $\alpha \equiv \vee(\exists \bigcup_{j=1}^n X'_j.\beta''_1[X_1 \leftarrow X'_1], \ldots, \exists \bigcup_{j=1}^n X'_j.\beta''_n[X_n \leftarrow X'_n]) \equiv \exists \bigcup_{j=1}^n X'_j.\vee(\beta''_1[X_1 \leftarrow X'_1], \ldots, \beta''_n[X_n \leftarrow X'_n])$. Since $\vee(\beta''_1[X_1 \leftarrow X'_1], \ldots, \beta''_n[X_n \leftarrow X'_n])$ is an $\mathcal{L}[\vee]$ representation, the conclusion follows. Note that the set of free variables of α is preserved by the translation.
  - If α is not an $\mathcal{L}$ representation and $\alpha = \exists x.\beta$ with $\beta \in \mathcal{L}[\vee, \exists]$, then by induction hypothesis, one can compute in polynomial time a representation $\beta' \in (\mathcal{L}[\vee])[\exists]$ such that $\beta' \equiv \beta$. Since $\exists x.\beta'$ is an $(\mathcal{L}[\vee])[\exists]$ representation equivalent to α, the conclusion follows. Again, the set of free variables of α is preserved by the translation.
- $\mathcal{L}[\vee, \exists] \geq_p (\mathcal{L}[\exists])[\vee]$. Again, the proof is by structural induction. Let α be any representation from $\mathcal{L}[\vee, \exists]$:
  - If α is an $\mathcal{L}$ representation, then it is also an $(\mathcal{L}[\exists])[\vee]$ representation due to the inclusion $(\mathcal{L}[\exists])[\vee] \supseteq \mathcal{L}$ which comes from item 0. in Proposition 1.
  - If $\alpha = \vee(\beta_1, \ldots, \beta_n)$ with $\beta_i \in \mathcal{L}[\vee, \exists]$ $(i \in 1, \ldots, n)$, then by induction hypothesis, one can compute in polynomial time n representations $\beta'_i \in (\mathcal{L}[\exists])[\vee]$ $(i \in 1, \ldots, n)$ such that for each $i \in 1, \ldots, n$, $\beta'_i \equiv \beta_i$. Since $\vee(\beta'_1, \ldots, \beta'_n)$ is an $(\mathcal{L}[\exists])[\vee]$ representation equivalent to α, the conclusion follows.
  - If $\alpha = \exists x.\beta$ with $\beta \in \mathcal{L}[\vee, \exists]$, then by induction hypothesis, one can compute in polynomial time a representation $\beta' \in (\mathcal{L}[\exists])[\vee]$ such that $\beta' \equiv \beta$. If β' is an $\mathcal{L}[\exists]$ representation, then $\exists x.\beta'$ also is an $\mathcal{L}[\exists]$ representation; since it is equivalent to α and since $(\mathcal{L}[\exists])[\vee] \supseteq \mathcal{L}[\exists]$ (see item 0. in Proposition 1), the conclusion follows. Otherwise we have $\beta' = \vee(\beta'_1, \ldots, \beta'_n)$ where β'_i is an $\mathcal{L}[\exists]$ representation $(i \in 1, \ldots, n)$. By replacement, α is equivalent to $\exists x.\vee(\beta'_1, \ldots, \beta'_n)$, which is equivalent to $\vee((\exists x.\beta'_1), \ldots, (\exists x.\beta'_n))$. Since the latter representation is an $(\mathcal{L}[\exists])[\vee]$ representation, the conclusion follows. □

Proposition 3. *Let $\mathcal{L}$ be any subset of C-*QDAG *s.t. $\mathcal{L}$ is stable by uniform renaming.*

- *If $\mathcal{L}$ satisfies* **CO** *(resp.* **CD***), then $\mathcal{L}[\vee]$, $\mathcal{L}[\exists]$ and $\mathcal{L}[\vee, \exists]$ satisfy* **CO** *(resp.* **CD***).*
- *If $\mathcal{L}$ satisfies* **CO** *and* **CD***, then $\mathcal{L}$ satisfies* **CE** *and* **ME***.*

[8] A key observation here is that all β_i $(i \in 1, \ldots, n)$ are pairwise independent, i.e., they do not share any node and for $i \in 1, \ldots, n$, every arc reaching a node of β_i comes from a node of β_i; if this was not the case, such a proof by structural induction would not work.

Algorithm 1: enumerate(α, γ).

input : an $\mathcal{L}$ representation α, and a set γ of literals over $Var(\alpha)$

```
1 if α is consistent then
2     if Var(α) = ∅ then
3         write(γ)
4     else
5         x ← first(Var(α))
6         enumerate(α_|x, γ ∪ {x})
7         enumerate(α_|¬x, γ ∪ {¬x})
```

- *If $\mathcal{L}$ satisfies* **CO** *and* **CD**, *then $\mathcal{L}$, $\mathcal{L}[\vee]$, $\mathcal{L}[\exists]$ and $\mathcal{L}[\vee, \exists]$ satisfy* **MC**.
- *$\mathcal{L}[\vee]$ and $\mathcal{L}[\vee, \exists]$ satisfy $\vee$***C** *(hence $\vee$***BC***) and $\mathcal{L}[\exists]$ and $\mathcal{L}[\vee, \exists]$ satisfy* **FO** *(hence* **SFO***).*
- *If $\mathcal{L}$ satisfies* **FO** *(resp.* **SFO***), then $\mathcal{L}[\vee]$ satisfies* **FO** *(resp.* **SFO***).*
- *If $\mathcal{L}$ satisfies $\wedge$***C** *(resp. $\wedge$***BC**, $\vee$**C**, $\vee$**BC***), then $\mathcal{L}[\exists]$ satisfies $\wedge$***C** *(resp. $\wedge$***BC**, $\vee$**C**, $\vee$**BC***).*

Proof.

- As to **CO**, since $\mathcal{L}[\vee] \subseteq \mathcal{L}[\vee, \exists]$ and $\mathcal{L}[\exists] \subseteq \mathcal{L}[\vee, \exists]$, it is enough to show that $\mathcal{L}[\vee, \exists]$ satisfies **CO**. Let α be any representation from $\mathcal{L}[\vee, \exists]$; since $\mathcal{L}[\vee, \exists] \sim_p (\mathcal{L}[\vee])[\exists]$ (cf. Proposition 2), we can compute in time polynomial in the size of α an equivalent representation $\beta = \exists X. \vee(\beta_1, \ldots, \beta_n)$ where X is a finite subset of PS and each β_i ($i \in 1, \ldots, n$) is an $\mathcal{L}$ representation. We have that α is consistent iff β is consistent iff $\vee(\beta_1, \ldots, \beta_n)$ is consistent iff at least one β_i ($i \in 1, \ldots, n$) is consistent. Since the latter can be decided in polynomial time, the conclusion follows.
  As to **CD**, let γ be any consistent term. Let α be an $\mathcal{L}[\vee]$ representation; we have $\alpha = \vee(\beta_1, \ldots, \beta_n)$ where each β_i ($i \in 1, \ldots, n$) is an $\mathcal{L}$ representation. Since $\wedge$ distributes over $\vee$ and existential quantifications "distribute" over $\vee$ as well, we have $\exists Var(\gamma).(\alpha \wedge \gamma) \equiv \exists Var(\gamma).(\vee(\beta_1, \ldots, \beta_n) \wedge \gamma) \equiv \exists Var(\gamma). \vee (\beta_1 \wedge \gamma, \ldots, \beta_n \wedge \gamma) \equiv \vee(\exists Var(\gamma).(\beta_1 \wedge \gamma), \ldots, \exists Var(\gamma).(\beta_n \wedge \gamma))$. If $\mathcal{L}$ satisfies **CD**, then each $\exists Var(\gamma).(\beta_i \wedge \gamma)$ ($i \in 1, \ldots, n$) can be associated in polynomial time with an equivalent $\mathcal{L}$ representation β'_i. Hence $\exists Var(\gamma).(\alpha \wedge \gamma)$ is equivalent to the $\mathcal{L}[\vee]$ representation $\vee(\beta'_1, \ldots, \beta'_n)$ which can be computed in time polynomial in the size of the input. Now let α be an $\mathcal{L}[\exists]$ representation; we have $\alpha = \exists X.\beta$ where X is a finite subset of PS and β is an $\mathcal{L}$ representation.
  We have $\exists Var(\gamma).(\alpha \wedge \gamma) \equiv \exists Var(\gamma[X \leftarrow X']).((\exists X.\beta) \wedge \gamma[X \leftarrow X'])$ where $\gamma[X \leftarrow X']$ is the representation obtained by replacing in γ every variable $x \in Var(\gamma) \cap X$ by a fresh variable x', not occurring in β or γ. Since $Var(\gamma[X \leftarrow X']) \cap X = \emptyset$, we have that $\exists Var(\gamma[X \leftarrow X']).((\exists X.\beta) \wedge \gamma[X \leftarrow X']) \equiv \exists Var(\gamma[X \leftarrow X']) \cup X.(\beta \wedge \gamma[X \leftarrow X']) \equiv \exists X.(\exists Var(\gamma[X \leftarrow X']).(\beta \wedge \gamma[X \leftarrow X']))$. If $\mathcal{L}$ satisfies **CD**, then $\exists Var(\gamma[X \leftarrow X']).(\beta \wedge \gamma[X \leftarrow X'])$ can be associated in polynomial time with an equivalent $\mathcal{L}$ representation β'. Hence $\exists Var(\gamma).(\alpha \wedge \gamma)$ is equivalent to the $\mathcal{L}[\exists]$ representation $\exists X.\beta'$ which can be computed in time polynomial in the size of the input. Finally, let α be an $\mathcal{L}[\vee, \exists]$ representation; since $\mathcal{L}[\vee, \exists] \sim_p (\mathcal{L}[\vee])[\exists]$ (cf. Proposition 2), we can compute in time polynomial in the size of α an equivalent representation $\beta = \exists X. \vee(\beta_1, \ldots, \beta_n)$ where X is a finite subset of PS and each β_i ($i \in 1, \ldots, n$) is an $\mathcal{L}$ representation. Then it is enough to combine the two previous proofs to get the desired result.
- We generalize some easy lemmata from [1] to the C-QDAG case. As to **CE**, it is enough to observe that for any C-QDAG representation α and any non-valid clause δ, we have $\alpha \models \delta$ iff $\alpha \wedge \neg\delta$ is inconsistent iff $\exists Var(\neg\delta).(\alpha \wedge \neg\delta)$ is inconsistent.
  As to **ME**, let α be any $\mathcal{L}$ representation. Procedure 1 enumerates the models of α over $Var(\alpha)$. It amounts to searching a decision tree T in a depth-first manner. Each branch of T corresponds either to a model of α over $Var(\alpha)$, or to an implicant of $\neg\alpha$. Each model is represented as a set of literals over $Var(\alpha)$. The procedure is called with $\gamma = \emptyset$. Given a total, strict ordering over the variables of $Var(\alpha)$, the function $first(\alpha)$ at Line 4 returns the first variable of α w.r.t. this ordering.
  Procedure 1 first consists in testing whether α is consistent (Line 1). If α is inconsistent, then the procedure stops; otherwise, one checks whether $Var(\alpha)$ is empty or not (Line 2). If this set is empty, then one returns the model of α stored in the accumulator γ (Line 3). In the remaining case, one computes the first variable x of α (Line 5). Afterwards, the procedure enumerates recursively all the models of $\alpha_{|x}$ by adding x to the accumulator γ (Line 6), then all the models of $\alpha_{|\neg x}$ by adding $\neg x$ to the accumulator γ (Line 7). In both cases, a variable is removed (since $x \notin Var(\alpha_{|x}) \cup Var(\alpha_{|\neg x})$), hence the number of recursive calls for each branch of T cannot exceed the number of variables of α. Furthermore, since $\mathcal{L}$ satisfies **CO** and **CD**, the time spent between two successive calls is polynomial in the input size. Procedure 1 is thus a polynomial delay model enumeration algorithm: a first model of α (when it exists) is generated in time polynomial in the size of the input, and after each model generation, the time needed to generate a further model (or to determine that no more models exist) also is polynomial in the size of the input. As a consequence, it runs in time polynomial in the size of the input plus the size of the output.
- Due to the inclusions $\mathcal{L} \subseteq \mathcal{L}[\vee]$, $\mathcal{L} \subseteq \mathcal{L}[\exists] \subseteq \mathcal{L}[\vee, \exists]$ (see Proposition 1), it is enough to show that $\mathcal{L}[\vee, \exists]$ satisfies **MC**. Let α be any $\mathcal{L}[\vee, \exists]$ representation. Since $\mathcal{L}[\vee, \exists] \sim_p (\mathcal{L}[\vee])[\exists]$ (cf. Proposition 2), we can compute in time polynomial in the size of α an equivalent representation $\beta = \exists X. \vee(\beta_1, \ldots, \beta_n)$ where X is a finite subset of PS and each β_i ($i \in 1, \ldots, n$) is an $\mathcal{L}$ representation. Furthermore, we have $Var(\beta) = Var(\alpha)$. Let ω be any interpretation over $Var(\alpha)$

and let γ be the consistent term (unique up to logical equivalence) such that $Var(\gamma) = Var(\alpha)$ and ω is a model of γ. We have $\omega \models \alpha$ iff $\gamma \wedge \beta$ is consistent iff $\gamma \wedge \vee(\beta_1, \ldots, \beta_n)$ is consistent iff there exists $i \in 1, \ldots, n$ such that $\gamma \wedge \beta_i$ is consistent iff there exists $i \in 1, \ldots, n$ such that $\exists Var(\gamma).(\beta_i \wedge \gamma)$ is consistent. Since $\mathcal{L}$ satisfies **CO** and **CD**, the conclusion follows.

- The fact that $\mathcal{L}[\vee]$ and $\mathcal{L}[\vee, \exists]$ satisfy $\vee$**C** and $\mathcal{L}[\exists]$ and $\mathcal{L}[\vee, \exists]$ satisfy **FO** is obvious (by construction).
- We prove the **FO** case (for **SFO** just take X as a singleton). Let α be a representation from $\mathcal{L}[\vee]$ and $X \subseteq PS$. By construction, $\alpha = \vee(\beta_1, \ldots, \beta_n)$ where each β_i $(i \in 1, \ldots, n)$ is an $\mathcal{L}$ representation. Since existential quantifications "distribute" over $\vee$, we have $\exists X.\alpha \equiv \vee(\exists X.\beta_1, \ldots, \exists X.\beta_n)$. Now, since $\mathcal{L}$ satisfies **FO**, with each $\exists X.\beta_i$ $(i \in 1, \ldots, n)$ we can associate in polynomial time an equivalent $\mathcal{L}$ representation β_i'. Applying the replacement metatheorem,[9] we get that $\exists X.\alpha \equiv \vee(\beta_1', \ldots, \beta_n')$. Since the $\mathcal{L}[\vee]$ representation $\vee(\beta_1', \ldots, \beta_n')$ can be computed in polynomial time in the size of α plus the size of X, the conclusion follows.
- We prove the $\wedge$**C** case. Let us consider n representations $\alpha_1, ..., \alpha_n$ from $\mathcal{L}[\exists]$ where $\mathcal{L}$ satisfies $\wedge$**C**. By construction, for each $i \in 1, \ldots, n$, α_i is of the form $\exists X_i.\beta_i$ where X_i is a finite subset of PS and $\beta_i \in \mathcal{L}$. With each $\exists X_i.\beta_i$ we can associate in polynomial time the equivalent representation $\exists X_i^i.\beta_i[X_i \leftarrow X_i^i]$ obtained by renaming in a uniform way every occurrence of variable $x \in X_i$ by the fresh variable x^i. Whenever β_i belongs to $\mathcal{L}$, $\beta_i[X_i \leftarrow X_i^i]$ belongs to $\mathcal{L}$ as well (due to the stability condition). From the replacement metatheorem, we get that $\bigwedge_{i=1}^n \alpha_i \equiv \bigwedge_{i=1}^n (\exists X_i.\beta_i) \equiv \bigwedge_{i=1}^n (\exists X_i^i.\beta_i[X_i \leftarrow X_i^i])$. By construction, we have $X_i^i \cap X_j^j = \emptyset$ when $i \neq j$. As a consequence, we have $\bigwedge_{i=1}^n (\exists X_i^i.\beta_i[X_i \leftarrow X_i^i]) \equiv \exists \bigcup_{i=1}^n X_i^i.(\bigwedge_{i=1}^n \beta_i[X_i \leftarrow X_i^i])$ Since $\mathcal{L}$ satisfies $\wedge$**C**, we can turn in polynomial time the representation $\bigwedge_{i=1}^n \beta_i[X_i \leftarrow X_i^i]$ into an equivalent representation β from $\mathcal{L}$. Since $\bigwedge_{i=1}^n \alpha_i \equiv \exists \bigcup_{i=1}^n X_i^i.\beta$ and $\bigcup_{i=1}^n X_i^i.\beta$ is an $\mathcal{L}[\exists]$ representation, the conclusion follows. The proof is similar for the remaining cases ($\wedge$**BC**, $\vee$**C**, $\vee$**BC**). □

Proposition 4.

- KROM $\sim_p$ KROM[$\exists$].
- KROM[$\vee$] $\sim_p$ KROM[$\vee, \exists$].
- AFF $\sim_p$ AFF[$\exists$].
- AFF[$\vee$] $\sim_p$ AFF[$\vee, \exists$].

Proof. These polynomial equivalences come easily from the fact that each of KROM and AFF satisfies **FO** (cf. Proposition 6), plus the fact that existential quantifications "distribute" over disjunctions. □

Proposition 5. *The results in Table 1 hold.*

	CO	VA	CE	IM	EQ	SE	CT	ME	MC
renH[$\vee, \exists$]	√	∘	√	∘	∘	∘	∘	√	√
K/H[$\vee, \exists$]	√	∘	√	∘	∘	∘	∘	√	√
HORN[$\vee, \exists$]	√	∘	√	∘	∘	∘	∘	√	√
AFF[$\vee$]	√	∘	√	∘	∘	∘	∘	√	√
renH[$\vee$]	√	∘	√	∘	∘	∘	∘	√	√
K/H[$\vee$]	√	∘	√	∘	∘	∘	∘	√	√
HORN[$\vee$]	√	∘	√	∘	∘	∘	∘	√	√
KROM[$\vee$]	√	∘	√	∘	∘	∘	∘	√	√
renH[$\exists$]	√	√	√	√	∘	∘	∘	√	√
K/H[$\exists$]	√	√	√	√	∘	∘	∘	√	√
HORN[$\exists$]	√	√	√	√	∘	∘	∘	√	√
AFF	√	√	√	√	√	√	√	√	√
renH	√	√	√	√	√	√	∘	√	√
K/H	√	√	√	√	√	√	∘	√	√
HORN	√	√	√	√	√	√	∘	√	√
KROM	√	√	√	√	√	√	∘	√	√

KROM, HORN, K/H, AFF, renH, their disjunction, existential and full disjunctive closures and the corresponding polynomial-time queries. √ means "satisfies" and ∘ means "does not satisfy unless P = NP."

[9] In classical propositional logic, this metatheorem states that if β is a subformula of a propositional formula α and β' is a formula equivalent to β, then the formula obtained by replacing in α the subformula β by β' is a formula equivalent to α [63] (this comes directly from the truth-functionality of the connectives); this metatheorem also holds for quantified formulae and can be generalized to the case of DAG-based representations (under some conditions); more precisely, given any node N of a C-QDAG representation α let β_N be the subgraph of α given by the set S_N of nodes of α reachable from N; if every arc of α having its extremity in $S_N \setminus \{N\}$ also has its origin in S_N, then for every C-QDAG representation β' equivalent to β, the C-QDAG representation obtained by removing in α every node and every arc of β, and redirecting the arcs entering N to the root of β' is a representation equivalent to α.

Proof.

CO It is well-known that each of KROM, HORN, renH, AFF satisfies **CO** (cf. [64–67,25]). Since deciding whether a C-QDAG representation is in KROM (resp. HORN) can be done in polynomial time, we get that K/H satisfies **CO**. Then point 1. of Proposition 3 allows to conclude that each of the $[\vee]$, $[\exists]$, and $[\vee,\exists]$ closures of those languages satisfies **CO** as well.

VA KROM, HORN, K/H and renH satisfy **VA** since they are subsets of CNF and CNF satisfies **VA**. AFF satisfies **VA** since it satisfies **CT** (indeed, an AFF formula α is valid if and only if it has 2^n models where n is the cardinality of $Var(\alpha)$).
As to renH[∃], K/H[∃] and HORN[∃], the results hold since each of these languages satisfies **IM**. Obviously, every subset $\mathcal{L}$ of C-QDAG which satisfies **IM** satisfies **VA** as well (indeed, $\alpha \in \mathcal{L}$ is valid iff it is implied by the term $\top$). Since the proof that each of renH[∃], K/H[∃] and HORN[∃] satisfies **IM** relies on the fact that HORN[∃] satisfies **VA**, it just remains to show it. This is easy since a formula α from HORN[∃] is valid if and only if its universal closure is valid. The fact that the validity problem for closed, prenex quantified Boolean formulae with a HORN matrix is in P [68] concludes the proof.
Finally, none of KROM[∨], HORN[∨], K/H[∨], renH[∨], AFF[∨], HORN[∨,∃], K/H[∨,∃], renH[∨,∃] satisfies **VA** unless P = NP since each of those languages includes DNF as a subset and DNF does not satisfy **VA** unless P = NP.

CE, **ME** The results come directly from the second item of Proposition 3, given that each of the sixteen languages considered here satisfies both **CO** and **CD**.

IM As to KROM, HORN, K/H, renH, and AFF, the results come from the fact that if a subset of C-QDAG satisfies **VA** and **CD**, then it satisfies **IM** (this slightly extends Lemma A.7 from [1] to C-QDAG representations).
Consider now the case of renH[∃], K/H[∃] and HORN[∃]. Since each of K/H[∃] and HORN[∃] is polynomially translatable into renH[∃], it is enough to prove the result for renH[∃]. We first show that the implicant problem for renH[∃] formulae can be reduced in polynomial time into the implicant problem for HORN[∃] formulae. Let $\exists X.\alpha$ be a renH[∃] formula such that α is a renH formula, and let γ be a term. Let V be any Horn renaming for α. We have $\gamma \models \exists X.\alpha$ iff $\gamma \Rightarrow (\exists X.\alpha)$ is valid.
Now, viewing V as a substitution, one can take advantage of the substitution metatheorem for propositional logic. This theorem (see e.g., [63]) states that for any propositional formula Σ and any substitution σ (a mapping which replaces each variable by a formula), if Σ is valid, then $\sigma(\Sigma)$ is valid. With $\Sigma = \gamma \Rightarrow (\exists X.\alpha)$ and $\sigma = V$, we get that if $\gamma \Rightarrow (\exists X.\alpha)$ is valid, then $V(\gamma \Rightarrow (\exists X.\alpha))$ is valid. Since for every formula β, $V(V(\beta)) \equiv \beta$, we also get that if $V(\gamma \Rightarrow (\exists X.\alpha))$ is valid, then $\gamma \Rightarrow (\exists X.\alpha)$ is valid. Altogether, we get that $\gamma \Rightarrow (\exists X.\alpha)$ is valid iff $V(\gamma \Rightarrow (\exists X.\alpha))$ is valid. Now, $V(\gamma \Rightarrow (\exists X.\alpha))$ is valid iff $V(\gamma) \Rightarrow V(\exists X.\alpha)$ is valid iff $V(\gamma) \models V(\exists X.\alpha)$.
Let ω be any interpretation over $Var(\alpha) \cup X$. Since for every variable x, $V(x)$ is equal to x or is equal to $\neg x$, $V(\omega)$ can be viewed as well as an interpretation over $Var(\alpha) \cup X$. We have $\omega \models V(\exists X.\alpha)$ iff $V(\omega) \models \exists X.\alpha$ (using the substitution theorem and the fact that for every formula β, $V(V(\beta)) \equiv \beta$) iff there exists an interpretation ω' over $Var(\alpha) \cup X$ such that $\omega' \models \alpha$ and $\forall y \in (Var(\alpha) \cup X) \setminus X$, $V(\omega)(y) = \omega'(y)$ (by definition of $\exists X.\alpha$) iff there exists an interpretation $V(\omega')$ over $Var(\alpha) \cup X$ such that $V(\omega') \models V(\alpha)$ and $\forall y \in (Var(\alpha) \cup X) \setminus X$, $V(V(\omega))(y) = V(\omega')(y)$. Since $V(V(\omega)) = \omega$, this is equivalent to state that ω is a model of $\exists X.V(\alpha)$. As a consequence, we have $V(\exists X.\alpha) \equiv \exists X.V(\alpha)$.
Accordingly, γ is an implicant of the renH[∃] formula $\exists X.\alpha$ iff the term $V(\gamma)$ is an implicant of the HORN[∃] formula $\exists X.V(\alpha)$. As explained above (see the **VA** point in the proof), the fact that HORN[∃] satisfies **CD** and **VA** shows that it satisfies **IM** as well. Given that a Horn renaming V for α can be computed in polynomial time given α, and that $V(\gamma)$ (resp. $V(\alpha)$) can be computed in polynomial time from γ (resp. α) once V has been computed, the fact that HORN[∃] satisfies **IM** shows that renH[∃] satisfies **IM** as well.
Finally, none of KROM[∨], HORN[∨], K/H[∨], renH[∨], AFF[∨], HORN[∨,∃], K/H[∨,∃], renH[∨,∃] satisfies **IM** unless P = NP, since none of them satisfies **VA** unless P = NP.

SE Determining whether a KROM (resp. HORN, K/H, renH) formula β is a logical consequence of a KROM (resp. HORN, K/H, renH) formula α amounts to determining whether every clause of β is a logical consequence of α. The fact that each of KROM, HORN, K/H and renH satisfy **CE** completes the proof for those four languages. As to AFF, determining whether an AFF formula β is a logical consequence of an AFF formula α amounts to determining whether every XOR-clause of β is a logical consequence of α. Now, a XOR-clause $l_1 \oplus \ldots \oplus l_n$ is a logical consequence of an AFF formula α if and only if the AFF formula $\alpha \wedge (l_1 \oplus \ldots \oplus l_n \oplus \top)$ is inconsistent. The fact that AFF satisfies **CO** concludes the proof for AFF.
As to renH[∃], K/H[∃] and HORN[∃], it is enough to prove the result for HORN[∃] since this language is included in the two remaining ones. Let α be a CNF formula over n variables $x_1, \ldots, x_n$. Let α' be the HORN formula obtained by replacing every positive literal x_i in α by the negative literal $\neg x'_i$ (where each x'_i is a fresh variable), conjoined with n additional clauses $\neg x_i \vee \neg x'_i$ $(i \in 1, \ldots, n)$. Let β' be the KROM formula $\bigwedge_{i=1}^{n}(x_i \vee x'_i)$. By construction, α is inconsistent iff $\alpha' \wedge \beta'$ is inconsistent iff $\alpha' \models \neg\beta'$. $\neg\beta'$ is equivalent to $\bigvee_{i=1}^{n}(\neg x_i \wedge \neg x'_i)$, which in turn is equivalent to the formula $\gamma' = \exists\{y_1, \ldots, y_n\}((\neg y_1 \vee \ldots \vee \neg y_n) \wedge \bigwedge_{i=1}^{n}((y_i \vee \neg x_i) \wedge (y_i \vee \neg x'_i)))$ (where each y_i is a fresh variable). The fact that α' and γ' are HORN[∃] formulae which can be computed in time polynomial in the size of α shows the coNP-hardness of the sentential entailment problem for HORN[∃] formulae and concludes the proof.
Finally, none of KROM[∨], HORN[∨], K/H[∨], renH[∨], AFF[∨], HORN[∨,∃], K/H[∨,∃], renH[∨,∃] satisfies **SE** unless P = NP since none of them satisfies **VA** unless P = NP; the fact that $\top$ is a formula from each of these languages and that $\alpha \in C$-QDAG is valid iff $\top \models \alpha$ concludes the proof.

EQ Each of KROM, HORN, K/H, renH, and AFF satisfies **EQ** since it satisfies **SE**.
As to renH[∃], K/H[∃] and HORN[∃]: for every formulae α' and γ' from C-QDAG we have that $\alpha' \models \gamma'$ iff $\alpha' \wedge \gamma' \equiv \alpha'$. Consider now the formulae α' and γ' used for proving that none of renH[∃], K/H[∃] and HORN[∃] satisfies **SE** unless P = NP (see the item **SE** in this proof). Since none of the y_i variables occurs in α', the formula $\alpha' \wedge \gamma'$ can be turned in linear time into the equivalent formula $\exists\{y_1, \ldots, y_n\}.(\alpha' \wedge ((\neg y_1 \vee \ldots \vee \neg y_n) \wedge \bigwedge_{i=1}^{n}((y_i \vee \neg x_i) \wedge (y_i \vee \neg x_i'))))$, which is a HORN[∃] formula. This concludes the proof.
Finally, none of KROM[∨], HORN[∨], K/H[∨], renH[∨], AFF[∨], HORN[∨, ∃], K/H[∨, ∃], renH[∨, ∃] satisfies **EQ** unless P = NP since none of them satisfies **VA** unless P = NP; the fact that $\top$ is a formula from each of these languages and that $\alpha \in C$-QDAG is valid iff $\top \equiv \alpha$ concludes the proof.

CT The result for AFF is proven in [69]. The results for all the remaining languages come from the fact that the language of negative Krom formulae (i.e., the set of all conjunctions of negative, binary clauses) is included into each language among KROM, HORN, K/H, renH, KROM[∨], HORN[∨], K/H[∨], renH[∨], HORN[∨, ∃], K/H[∨, ∃], renH[∨, ∃]; furthermore, DNF is included in AFF[∨] since each term is an AFF formula. The fact that none of the language of negative Krom formulae and DNF satisfies **CT** [70] concludes the proof.

MC The results come directly from the third item of Proposition 3, given that each of KROM, HORN, K/H, renH, and AFF satisfies both **CO** and **CD**. □

Proposition 6. *The results in Table 2 hold.*

	CD	FO	SFO	EN	SEN	∧C	∧BC	∨C	∨BC	¬C
renH[∨, ∃]	√	√	√	○	○	○	○	√	√	○
K/H[∨, ∃]	√	√	√	○	√	○	○	√	√	○
HORN[∨, ∃]	√	√	√	○	√	○	√	√	√	○
AFF[∨]	√	√	√	○	√	○	√	√	√	○
renH[∨]	√	•	√	○	○	○	○	√	√	○
K/H[∨]	√	•	√	○	√	○	○	√	√	•
HORN[∨]	√	•	√	○	√	○	√	√	√	•
KROM[∨]	√	√	√	○	√	○	√	√	√	•
renH[∃]	√	√	√	√	√	!	!	!	!	!
K/H[∃]	√	√	√	√	√	!	!	!	!	!
HORN[∃]	√	√	√	√	√	√	√	!	!	!
AFF	√	√	√	√	√	√	√	!	!	!
renH	√	•	√	√	√	!	!	!	!	!
K/H	√	•	√	√	√	!	!	!	!	!
HORN	√	•	√	√	√	√	√	!	!	!
KROM	√	√	√	√	√	√	√	!	!	!

KROM, HORN, K/H, AFF, renH, their disjunction, existential and full disjunctive closures and the corresponding polynomial-time transformations. √ means "satisfies," • means "does not satisfy," while ○ means "does not satisfy unless P = NP." ! means that the transformation is not always feasible within the language.

Proof.

CD When α is a CNF formula and γ is a term, a CNF formula β equivalent to $\alpha_{|\gamma}$ can be computed in time polynomial in the size of α plus the size of γ by removing from α every clause containing a literal l from γ while removing the complementary literal $\bar{l}$ from every clause of α containing it. Obviously enough, removing clauses and shortening clauses are two internal laws in the languages KROM and HORN. This shows that KROM, HORN and K/H satisfy **CD**. Similarly, when α is an AFF formula and γ is a term (viewed as a set of literals), an AFF formula β equivalent to $\alpha_{|\gamma}$ can be computed in time polynomial in the size of α plus the size of γ by replacing in α every occurrence of a literal l by $\top$ when l belongs to γ and by $\bot$ when $\bar{l}$ belongs to γ. As to renH, it is not hard to see that if V is a Horn renaming for a renH formula α then for any term γ, V also is Horn renaming for the formula β as defined above. Hence renH also satisfies **CD**.
Then point 1. of Proposition 3 allows to conclude that each of the [∨], [∃], and [∨, ∃] closures of those languages satisfies **CD** as well.

FO Each of HORN[∃], K/H[∃], renH[∃], HORN[∨, ∃], K/H[∨, ∃], renH[∨, ∃] obviously satisfies **FO** since such a transformation can be done in an implicit way in each of those languages.
As to KROM, it is well-known that the set of prime implicates of a KROM formula α can be computed in time polynomial in the size of α and that each such prime implicate is a binary clause (see [71]). Furthermore, the prime implicates of $\exists X.\alpha$ with $X \subseteq PS$ are the prime implicates of α which do not contain any atom from X (Proposition 55 in [71]), showing in particular that PI satisfies **FO**. Together, this shows that KROM satisfies **FO**.
The fact that AFF satisfies **FO** is given by Lemma 1 from [72].
Now, taking advantage of the fact that for any C-QDAG representation of the form $\vee(\alpha_1, \ldots, \alpha_n)$ and any finite subset X of PS $\exists X. \vee(\alpha_1, \ldots, \alpha_n)$ is logically equivalent to $\vee(\exists X.\alpha_1, \ldots, \exists X.\alpha_n)$, we get that each of KROM[∨] and AFF[∨] satisfies **FO**.

It remains to consider the cases of HORN, K/H, renH and of their disjunction closures. Consider the HORN formula $\alpha_n = (\bigvee_{i=1}^{n} \neg x_i) \wedge \bigwedge_{i=1}^{n} (x_i \vee \neg y_i) \wedge (x_i \vee \neg z_i)$ and the set $X_n = \{x_1, \ldots, x_n\}$ of atoms. Every clause of the form $\bigvee_{i=1}^{n} \neg l_i$ where l is y or z is an essential prime implicate of $\exists X_n.\alpha_n$ and there are 2^n such clauses. This shows that $\exists X_n.\alpha_n$ has only exponential size CNF representations. Thus HORN does not satisfy **FO**. Since α_n also is a K/H formula and a renH formula, we also get that none of K/H and renH satisfies **FO**.

Finally, the fact that HORN[$\vee$] (resp. K/H[$\vee$], renH[$\vee$]) does not satisfy **FO** comes from the fact that HORN[$\vee, \exists$] $<_s$ HORN[$\vee$] (resp. K/H[$\vee, \exists$] $<_s$ K/H[$\vee$], renH[$\vee, \exists$] $<_s$ renH[$\vee$]). Let us consider the HORN case (the other cases are similar): forgetting a set of variables X in a HORN[$\vee$] formula α amounts to computing a HORN[$\vee$] formula equivalent to the HORN[$\vee$][$\exists$] formula $\exists X.\alpha$. If HORN[$\vee$] would satisfy **FO**, then every HORN[$\vee$][$\exists$] formula $\exists X.\alpha$ could be turned in polynomial time into an equivalent HORN[$\vee$] formula. Since HORN[$\vee$][$\exists$] $\sim_p$ HORN[$\vee, \exists$], we would have HORN[$\vee, \exists$] $\geq_p$ HORN[$\vee$]. But this conflicts with the fact that HORN[$\vee$] $\not\leq_s$ HORN[$\vee, \exists$] (in a nutshell, if no polynomial-space translation exists, then no polynomial-time translation exists).

SFO Obviously, every language satisfying **FO** satisfies **SFO** as well. Hence it is enough to consider the cases of HORN, K/H, renH and of their disjunction closures.

Let us consider first the HORN and renH cases. For any CNF formula α (viewed as a set of clauses) and a propositional variable $x \in PS$, one can compute from α in polynomial time the following three sets of clauses α^*, α^+, and α^-: first remove from α every valid clause to get a set of clauses α'; now, compute α^* as the set of clauses of α' not containing x as a variable, α^+ as the set of clauses of α' containing x as a (positive) literal, from which x is removed, and compute α^- as the set of clauses of α' containing $\neg x$ as a (negative) literal from which $\neg x$ is removed. By construction, the conjunction β of clauses from $\alpha^* \cup \{\delta^+ \vee \delta^- \mid \delta^+ \in \alpha^+, \delta^- \in \alpha^-\}$ is a CNF formula equivalent to $(\alpha \mid \neg x) \vee (\alpha \mid x)$, hence equivalent to $\exists x.\alpha$. Since none of α^+ and α^- can contain more clauses or more literals than α, it comes that β can be computed in time polynomial in the size of α. It remains to show that if α is HORN (resp. renH) then the corresponding β is HORN (resp. renH). Assume that α is HORN. Then every clause from α^* is a Horn clause; furthermore, by construction every clause $\delta^+ \in \alpha^+$ is a negative clause and every clause $\delta^- \in \alpha^-$ is a Horn clause; hence, every clause of the form $\delta^+ \vee \delta^-$ is a Horn clause. Similarly, if α is renH and V is any Horn renaming for it, then V also is a Horn renaming for the corresponding β. Hence HORN and renH satisfy **SFO**.

Since both KROM and HORN satisfy **SFO**, K/H satisfies **SFO** as well.

Finally, given that for any C-QDAG representation α and any atom $x \in PS$, we have $\exists x.\alpha \equiv (\exists x.(\alpha \wedge \neg x) \vee \exists x.(\alpha \wedge x))$, the results for HORN[$\vee$], K/H[$\vee$], and renH[$\vee$] come that each of these languages satisfies **CD** and $\vee$**BC**.

EN For any C-QDAG representations α and β and any finite subset X of PS we have the equivalence $\forall X.(\alpha \wedge \beta) \equiv (\forall X.\alpha) \wedge (\forall X.\beta)$. Furthermore, when δ is a clause, $\forall X.\delta$ is equivalent to the clause obtained by removing from δ every literal l such that $var(l) \in X$. Since removing literals from a KROM (resp. HORN) clause leads to a KROM (resp. HORN) clause, altogether we get that each of KROM and HORN satisfies **EN**, and this shows that K/H satisfies **EN** as well. Now, if α is a renH formula and V is a Horn renaming for it, then the formula obtained by removing in every clause of α every literal built up from a variable of X still is a renH formula (indeed, V is still a Horn renaming for it). Hence, renH also satisfies **EN**. Let us consider now the case of an AFF formula α. We assume w.l.o.g. that α is simplified, i.e., for every XOR-clause $\delta = l_1 \oplus \ldots \oplus l_k$ of α, either δ reduces to $\bot$, or every literal in δ is positive or equal to $\top$ and δ does not contain more than one occurrence of any variable and of $\top$ (if this is not the case it is sufficient to exploit the equivalences $\neg x \equiv x \oplus \top$, $\beta \oplus \beta \equiv \bot$, $\beta \oplus \bot \equiv \beta$ to render α simplified while preserving logical equivalence); it is easy to check that if α is a simplified AFF formula containing a variable from X, then $\forall X.\alpha$ is equivalent to $\bot$, otherwise $\forall X.\delta$ is equivalent to α. Hence, AFF satisfies **EN**.

The fact that HORN[$\exists$] satisfies **EN** is a consequence of Corollary 11 from [73]. Since KROM[$\exists$] $\sim_p$ KROM and KROM satisfies **EN**, as a consequence, we also have that K/H[$\exists$] satisfies **EN**.

As to the case of renH[$\exists$], let us consider a renH[$\exists$] formula $\alpha = \exists X.\beta$. Let V be a Horn renaming for β. Since HORN[$\exists$] satisfies **EN**, for every finite subset Y of PS, the formula $\forall Y.(\exists X.V(\beta))$ can be turned in polynomial time into an equivalent formula $\exists Z.\gamma$ from HORN[$\exists$]. From the substitution metatheorem, we have $V(\forall Y.(\exists X.V(\beta))) \equiv V(\exists Z.\gamma)$. Hence, we have $\forall Y.(\exists X.V(V(\beta))) \equiv \exists Z.V(\gamma)$. Since $V(V(\beta)) = \beta$, we get that $\forall Y.(\exists X.\beta) \equiv \exists Z.V(\gamma)$. Clearly, $\exists Z.V(\gamma)$ is a renH[$\exists$] formula; indeed, $V(\gamma)$ is a renH formula since $V(V(\gamma)) = \gamma$ is a HORN formula. Since $\exists Z.V(\gamma)$ can be computed in polynomial time from $\forall Y.(\exists X.\beta)$, we get that renH[$\exists$] satisfies **EN**.

Finally, for any C-QDAG representation α and any finite subset X of PS we have that α is valid iff $\forall Var(\alpha).\alpha$ is valid iff $\forall Var(\alpha).\alpha$ is consistent (since $\forall Var(\alpha).\alpha$ has no free variable, it is equivalent to $\top$ or to $\bot$, hence it is consistent precisely when it is valid). Hence every language satisfying **CO** but not satisfying **VA** unless $\mathsf{P} = \mathsf{NP}$ cannot satisfy **EN** unless $\mathsf{P} = \mathsf{NP}$. This is the case for each language among KROM[$\vee$], HORN[$\vee$], K/H[$\vee$], renH[$\vee$], AFF[$\vee$], HORN[$\vee, \exists$], K/H[$\vee, \exists$], renH[$\vee, \exists$].

SEN Every language satisfying **EN** also satisfies **SEN**. Hence, each of KROM, HORN, K/H, renH, HORN[$\exists$], K/H[$\exists$], renH[$\exists$], AFF satisfies **SEN**. Furthermore, since for any C-QDAG representation α and a variable $x \in PS$, we have $\forall x.\alpha \equiv \alpha_{|x} \wedge \alpha_{|\neg x}$, every language satisfying both **CD** and $\wedge$**BC** also satisfies **SEN**. Hence each of HORN[$\vee$], KROM[$\vee$], HORN[$\vee, \exists$] satisfies **SEN**. Since each of HORN[$\vee$], KROM[$\vee$] satisfies **SEN**, we also have that K/H[$\vee$] satisfies **SEN**. Similarly, since each of HORN[$\vee, \exists$], KROM[$\vee, \exists$] ($\sim_p$ KROM[$\vee$]) satisfies **SEN**, we have that K/H[$\vee, \exists$] satisfies **SEN**.

Finally, as to renH[$\vee$] and renH[$\vee, \exists$], let α be a CNF formula over n variables $x_1, \ldots, x_n$. Let α' be the HORN formula obtained by replacing every positive literal x_i in α by the negative literal $\neg x'_i$ (where each x'_i is a fresh

variable), conjoined with n additional clauses $\neg x_i \vee \neg x'_i$ ($i \in 1, \ldots, n$). Let β' be the KROM formula $\bigwedge_{i=1}^{n}(x_i \vee x'_i)$. By construction, α is inconsistent iff $\alpha' \wedge \beta'$ is inconsistent. Now, we associate α in polynomial time with the renH[$\vee$] formula $\gamma = (\alpha' \wedge \neg y) \vee (\beta' \wedge y)$ where y is a fresh variable. γ also is a renH[$\vee, \exists$] formula. We can easily check that $\forall y.\gamma$ is equivalent to $\alpha' \wedge \beta'$. If renH[$\vee$] (resp. renH[$\vee, \exists$]) would satisfy **SEN**, then we could compute in time polynomial in the size of α a renH[$\vee$] (resp. renH[$\vee, \exists$]) formula equivalent to $\forall y.\gamma$. Since each of renH[$\vee$] and renH[$\vee, \exists$] satisfies **CO**, we would have a polynomial time algorithm for deciding the satisfiability of α, hence we would have P = NP.

$\wedge$C It is obvious that each of KROM, HORN, and AFF satisfies **$\wedge$C**.
For K/H, renH, K/H[$\exists$], renH[$\exists$], the non-representability results (!) holds already in the bounded case (**$\wedge$BC**).
For K/H[$\vee$], renH[$\vee$], K/H[$\vee, \exists$], renH[$\vee, \exists$], the results comes from the fact that none of these languages satisfies **$\wedge$BC**, unless P = NP.
Consider now the cases of KROM[$\vee$], HORN[$\vee$], AFF[$\vee$] and HORN[$\vee, \exists$]. Observe that every clause is a formula from any of those languages since every literal is a KROM formula, a HORN formula, and an AFF formula. Determining whether a conjunction of clauses is consistent cannot be achieved in (deterministic) polynomial time unless P = NP (this is the famous SAT problem). Since each of KROM[$\vee$], HORN[$\vee$], AFF[$\vee$] and HORN[$\vee, \exists$] satisfies **CO**, none of them can also satisfy **$\wedge$C** unless P = NP.
Finally, let us consider the case of HORN[$\exists$]: let $\exists X_1.\alpha_1, \ldots, \exists X_n.\alpha_n$ be nHORN[$\exists$] formulae where each α_i ($i \in 1, \ldots, n$) is a HORN formula. For each $i \in 1, \ldots, n$, let α_i^i be the HORN formula obtained by replacing in α_i every occurrence of $x \in X_i$ by a fresh variable x^i, and let X_i^i be the set of all the variables x^i generated in the construction of α_i^i. By construction, every variable from X_i^i does not occur in any α_j^j when $j \neq i$. Hence, $\bigwedge_{i=1}^{n} \exists X_i.\alpha_i$ is equivalent to $\exists \bigcup_{i=1}^{n} X_i^i . \bigwedge_{i=1}^{n} \alpha_i^i$. Clearly enough, $\exists \bigcup_{i=1}^{n} X_i^i . \bigwedge_{i=1}^{n} \alpha_i^i$ is a HORN[$\exists$] formula, and it can be generated in polynomial time from $\exists X_1.\alpha_1, \ldots, \exists X_n.\alpha_n$.

$\wedge$BC Each of KROM, HORN, AFF and HORN[$\exists$] satisfies **$\wedge$BC** since it satisfies **$\wedge$C**.
As to K/H and K/H[$\exists$], consider the K/H formulae $x \vee y$ and $\neg x \vee \neg y \vee \neg z$. They are also K/H[$\exists$] formulae. The conjunction of them neither is equivalent to a KROM formula, nor is equivalent to a HORN formula, hence it is not equivalent to a K/H formula. From Proposition 7, we know that K/H $\sim_e$ K/H[$\exists$], hence this conjunction is not equivalent to a K/H[$\exists$] formula.
As to renH and renH[$\exists$], consider the two renH formulae $\alpha = x \vee y \vee z$ and $\beta = \neg x \vee \neg y \vee \neg z$. They are also renH[$\exists$] formulae. There is no renH formula logically equivalent to the conjunction $\alpha \wedge \beta$. From Proposition 7, we know that renH $\sim_e$ renH[$\exists$], hence this conjunction is not equivalent to a renH[$\exists$] formula.
Let us now consider the cases of KROM[$\vee$], HORN[$\vee$], and AFF[$\vee$]. Let $\alpha = \vee(\alpha_1, \ldots, \alpha_n)$ and $\beta = \vee(\beta_1, \ldots, \beta_m)$ be two KROM[$\vee$] (resp. HORN[$\vee$], AFF[$\vee$]) formulae. Then the formula $\bigvee_{i=1}^{n} \bigvee_{j=1}^{m} (\alpha_i \wedge \beta_j)$ can be computed in time polynomial in the size of α plus the size of β, and is a KROM[$\vee$] (resp. HORN[$\vee$], AFF[$\vee$]) formula logically equivalent to the conjunction $\alpha \wedge \beta$.
Let us focus on the case of HORN[$\vee, \exists$]. Let α and β be two HORN[$\vee, \exists$] formulae. From Proposition 2, since HORN is stable by uniform renaming, one can compute in polynomial time a HORN[$\vee$][$\exists$] formula $\exists X.\alpha'$ (resp. $\exists Y.\beta'$) equivalent to α (resp. β) where α' (resp. β') is a HORN[$\vee$] formula. Let α'' (resp. β'') be the HORN[$\vee$] formula obtained by replacing in α' (resp. β') every occurrence of $x \in X$ (resp. $x \in Y$) by a fresh variable x' and let X' (resp. Y') be the set of all variables x' generated in the construction of α'' (resp. β''). By construction $\alpha \wedge \beta$ is equivalent to $(\exists X.\alpha') \wedge (\exists Y.\beta')$, which is in turn equivalent to $\exists X' \cup Y'.(\alpha'' \wedge \beta'')$. Now, HORN[$\vee$] satisfies **$\wedge$BC**. Hence, a HORN[$\vee$] formula γ equivalent to $\alpha'' \wedge \beta''$ can be generated in time polynomial in the size of α'' plus the size of β''. Accordingly, $\exists X' \cup Y'.\gamma$ is a HORN[$\vee, \exists$] formula equivalent to $\alpha \wedge \beta$, and it can be computed in time polynomial in the size of α plus the size of β.
Finally, let us consider the cases of K/H[$\vee$], renH[$\vee$], K/H[$\vee, \exists$], renH[$\vee, \exists$]. Let α be a CNF formula over n variables $x_1, \ldots, x_n$. Let α' be the HORN formula obtained by replacing every positive literal x_i in α by the negative literal $\neg x'_i$ (where each x'_i is a fresh variable), conjoined with n additional clauses $\neg x_i \vee \neg x'_i$ ($i \in 1, \ldots, n$). Let β' be the KROM formula $\bigwedge_{i=1}^{n}(x_i \vee x'_i)$. Observe that β' is consistent, hence each of α' and β' is a K/H formula and renH formula. As a consequence, each of them also belongs to K/H[$\vee$], renH[$\vee$], K/H[$\vee, \exists$], and renH[$\vee, \exists$]. Furthermore, both α' and β' can be computed in time polynomial in the size of α. By construction, α is consistent iff $\alpha' \wedge \beta'$ is consistent. If any of K/H[$\vee$], renH[$\vee$], K/H[$\vee, \exists$], or renH[$\vee, \exists$] would satisfy **$\wedge$BC**, since each of these languages satisfy **CO**, we would have P = NP.

$\vee$C The non-representability results for KROM, HORN, K/H, renH, AFF, HORN[$\exists$], K/H[$\exists$], renH[$\exists$] come directly from the corresponding non-representability results for **$\vee$BC**. The fact that each of KROM[$\vee$], HORN[$\vee$], K/H[$\vee$], renH[$\vee$], AFF[$\vee$], HORN[$\vee, \exists$], K/H[$\vee, \exists$], renH[$\vee, \exists$] satisfies **$\vee$C** is immediate from their definitions.

$\vee$BC Consider the two KROM formulae $\alpha = (x \vee y) \wedge (\neg y \vee \neg z)$ and $\beta = \neg x \wedge z$. Each of α and β belongs as well to K/H, renH, K/H[$\exists$], renH[$\exists$]. Now, $\alpha \vee \beta$ is logically equivalent to the formula $(x \vee y \vee z) \wedge (\neg x \vee \neg y \vee \neg z)$ for which no equivalent KROM formula (resp. K/H formula, renH formula) exists. From Proposition 7, we know that KROM $\sim_e$ KROM[$\exists$] (resp. K/H $\sim_e$ K/H[$\exists$], renH $\sim_e$ renH[$\exists$]), hence there are no KROM[$\exists$] formula (resp. K/H[$\exists$] formula, renH[$\exists$] formula) equivalent to $\alpha \vee \beta$.

As to the HORN case and the AFF case, it is enough to consider $\alpha = x$ and $\beta = y$: no HORN formula and no AFF formula is equivalent to $\alpha \vee \beta$. From Proposition 7, we know that HORN $\sim_e$ HORN[$\exists$], hence there is no HORN[$\exists$] formula equivalent to $\alpha \vee \beta$.
Finally, the fact that each of KROM[$\vee$], HORN[$\vee$], K/H[$\vee$], renH[$\vee$], AFF[$\vee$], HORN[$\vee, \exists$], K/H[$\vee, \exists$], renH[$\vee, \exists$] satisfies $\vee$**BC** comes from the fact that each of them satisfies $\vee$**C**.

¬C Consider the KROM formula $\alpha = (\neg x \vee y) \wedge (\neg x \vee z) \wedge (x \vee \neg y) \wedge (\neg y \vee z) \wedge (x \vee \neg z) \wedge (y \vee \neg z)$. α also is a HORN formula, a K/H formula, a renH formula, a HORN[$\exists$] formula, a K/H[$\exists$] formula, and a renH[$\exists$] formula. But $\neg\alpha$ is equivalent to the formula $(x \vee y \vee z) \wedge (\neg x \vee \neg y \vee \neg z)$ for which no equivalent KROM formula (resp. HORN formula, K/H formula, renH formula) exists. From Proposition 7, we know that HORN $\sim_e$ HORN[$\exists$], K/H $\sim_e$ K/H[$\exists$], and renH $\sim_e$ renH[$\exists$]. Accordingly, the **¬C** transformation is not always feasible in any of KROM, HORN, K/H, renH, HORN[$\exists$], K/H[$\exists$], renH[$\exists$].
Similarly, consider the AFF formula $\alpha = \neg x \wedge \neg y$. No AFF formula is equivalent to $\neg\alpha$, hence the **¬C** transformation is not always feasible in AFF.
As to KROM[$\vee$], HORN[$\vee$], K/H[$\vee$], let us consider the DNF formula $\alpha_n = \bigvee_{i=1}^{n}(\neg x_i \wedge \neg y_i \wedge \neg z_i)$; α_n is a KROM[$\vee$] formula, a HORN[$\vee$] formula and a K/H[$\vee$] formula. Now, in the proof of Proposition 11 (see Table 9), we show that the renH formula $\bigwedge_{i=1}^{n}(x_i \vee y_i \vee z_i)$ equivalent to $\neg\alpha_n$ has no polynomial-size representation in K/H[$\vee$], hence the conclusion follows.
Finally, let us consider the cases of AFF[$\vee$], renH[$\vee$], renH[$\vee, \exists$], K/H[$\vee, \exists$] and HORN[$\vee, \exists$]. DNF is a subset of each of these languages. Now, a DNF formula α is valid iff $\neg\alpha$ is inconsistent. The fact that each of AFF[$\vee$], renH[$\vee$], renH[$\vee, \exists$], K/H[$\vee, \exists$] and HORN[$\vee, \exists$] satisfies **CO** completes the proof. □

Proposition 7.

- HORN[$\exists$] $\sim_e$ HORN.
- K/H[$\exists$] $\sim_e$ K/H.
- renH[$\exists$] $\sim_e$ renH.

Proof.

- HORN, K/H: Every prime implicate of a HORN formula (resp. a KROM formula) α is a Horn clause (resp. a binary clause). Since the prime implicates of $\exists X.\alpha$ for a finite subset X of PS and a C-DAG representation α are the prime implicates δ of α such that $Var(\delta) \cap X = \emptyset$, we get that $\exists X.\alpha$ is equivalent to a HORN formula (resp. a KROM formula) when α is a HORN formula (resp. a KROM formula).
- renH: Let α be a renH formula. A PI formula equivalent to $\exists X.\alpha$ is given by the conjunction β of all prime implicates of α not containing any variable from X. If V is a Horn renaming for α, then $V(\alpha)$ is a HORN formula. Since $V(\beta)$ is equivalent to $\exists X.V(\alpha)$ and since HORN[$\exists$] $\sim_e$ HORN, $V(\beta)$ is equivalent to a HORN formula. This shows that β is a renH formula (V is a Horn renaming for it) and this concludes the proof. □

Proposition 8. KROM[$\vee$], HORN[$\vee$], K/H[$\vee$], renH[$\vee$], AFF[$\vee$], HORN[$\vee, \exists$], K/H[$\vee, \exists$], renH[$\vee, \exists$] *are complete propositional languages.*

Proof. This comes easily from the fact that TERM is included in each of KROM, HORN and AFF; as a consequence, its disjunction closure TERM[$\vee$] is included into each of eight closures above; the fact that DNF = TERM[$\vee$] is complete ends up the proof. □

Proposition 9.

- HORN[$\exists$] $<_s$ HORN.
- K/H[$\exists$] $<_s$ K/H.
- renH[$\exists$] $<_s$ renH.
- renH *and* K/H[$\exists$] *are incomparable w.r.t.* $\leq_s$.
- K/H *and* HORN[$\exists$] *are incomparable w.r.t.* $\leq_s$.

Proof. Let us consider first the three first items. For $\mathcal{L} \in \{$HORN, K/H, renH$\}$, we have to prove that $\mathcal{L}[\exists] <_s \mathcal{L}$, i.e., $\mathcal{L}[\exists] \leq_s \mathcal{L}$ and $\mathcal{L} \not\leq_s \mathcal{L}[\exists]$. That $\mathcal{L}[\exists] \leq_s \mathcal{L}$ comes immediately from the inclusion $\mathcal{L}[\exists] \supseteq \mathcal{L}$ (cf. item 0. of Proposition 1). The other way around, consider the HORN[$\exists$] formula $\alpha_n = \exists\{y_1, \ldots, y_n\}.((\bigvee_{i=1}^{n} \neg y_i) \wedge \bigwedge_{i=1}^{n}((\neg x_i \vee y_i) \wedge (\neg z_i \vee y_i)))$. Since HORN $\subseteq$ K/H and HORN $\subseteq$ renH, this is also a K/H[$\exists$] formula and a renH[$\exists$] formula (cf. item 0. of Proposition 1). Since α_n has 2^n essential prime implicates, it does not have a CNF representation of size polynomial in n. Since HORN, K/H, and renH are subsets of CNF, the language CNF is at least as succinct as any of them, so α_n does not have a representation of size of polynomial in n as a HORN formula, a K/H formula or a renH formula.

For the last two items, we have to prove that renH $\not\leq_s$ K/H[$\exists$], K/H[$\exists$] $\not\leq_s$ renH, K/H $\not\leq_s$ HORN[$\exists$], HORN[$\exists$] $\not\leq_s$ K/H. From Proposition 7, we know that K/H[$\exists$] $\sim_e$ K/H, and that HORN[$\exists$] $\sim_e$ HORN. Furthermore, we know that renH $<_e$ K/H $<_e$ HORN (cf. Section 5). Altogether, this shows that renH $<_e$ K/H[$\exists$] and both $<_e$ HORN[$\exists$]. Especially, we have K/H[$\exists$] $\not\leq_e$ renH and HORN[$\exists$] $\not\leq_e$ K/H. Due to the fact that the relation $\leq_s$ is included into the relation $\leq_e$, we have that for any subsets $\mathcal{L}_1$ and $\mathcal{L}_1$ of C-QDAG, if $\mathcal{L}_1 \not\leq_e \mathcal{L}_2$, then $\mathcal{L}_1 \not\leq_s \mathcal{L}_2$. This shows that K/H[$\exists$] $\not\leq_s$ renH and HORN[$\exists$] $\not\leq_s$ K/H. Finally, in order to prove that renH $\not\leq_s$ K/H[$\exists$] and K/H $\not\leq_s$ HORN[$\exists$], it is enough to consider again the horn[$\exists$] formula $\alpha_n = \exists\{y_1,\ldots,y_n\}.((\bigvee_{i=1}^n \neg y_i) \wedge \bigwedge_{i=1}^n((\neg x_i \vee y_i) \wedge (\neg z_i \vee y_i)))$. We have shown above that this formula also is a K/H[$\exists$] formula but that it does not have a representation of size of polynomial in n as a renH formula or as a K/H formula. This concludes the proof. □

Proposition 10.

- HORN[$\vee,\exists$] $<_s$ HORN[$\vee$].
- K/H[$\vee,\exists$] $<_s$ K/H[$\vee$].
- renH[$\vee,\exists$] $<_s$ renH[$\vee$].

Proof. We focus on AC3, the class of propositional representations containing all disjunctions of CNF formulae and all conjunctions of DNF formulae. Since every formula from HORN[$\vee$], K/H[$\vee$] or renH[$\vee$] is a disjunction of CNF formulae, each of the languages HORN[$\vee$], K/H[$\vee$], and renH[$\vee$] is a subset of AC3, hence we have AC3 $\leq_s$ HORN[$\vee$], AC3 $\leq_s$ K/H[$\vee$], and AC3 $\leq_s$ renH[$\vee$]. In order to prove the proposition, it is thus enough to show that AC3 $\not\leq_s$ HORN[$\vee,\exists$], AC3 $\not\leq_s$ K/H[$\vee,\exists$], and AC3 $\not\leq_s$ renH[$\vee,\exists$]. Since HORN $\subseteq$ K/H and HORN $\subseteq$ renH, we have the inclusions HORN[$\vee,\exists$] $\subseteq$ K/H[$\vee,\exists$] and HORN[$\vee,\exists$] $\subseteq$ renH[$\vee,\exists$] (cf. item 0. of Proposition 1), which imply that K/H[$\vee,\exists$] $\leq_s$ HORN[$\vee,\exists$], and renH[$\vee,\exists$] $\leq_s$ HORN[$\vee,\exists$]. Therefore, in order to show that AC3 $\not\leq_s$ HORN[$\vee,\exists$], AC3 $\not\leq_s$ K/H[$\vee,\exists$], and AC3 $\not\leq_s$ renH[$\vee,\exists$], it is enough to show that AC3 $\not\leq_s$ HORN[$\vee,\exists$]. We do it by exhibiting a HORN[$\vee,\exists$] formula which has no polynomial-sized AC3 representation.

The proof is based on a theorem due to Sipser [74]. This theorem can be expressed as follows: consider any Boolean function α_k^n over n^{2k-2} variables, represented by a NNF formula of depth $k>1$ and such that all the leaves are labeled by variables occurring once in the formula, the ith level ($i \in 1,\ldots,k-1$) from the bottom consists of nodes labeled by $\wedge$ (resp. $\vee$) when i is even (resp. odd), the outdegree of the root node and the deepest internal nodes (those at depth $k-1$) is equal to $n>1$ and the outdegree of every other internal node is equal to n^2. Sipser showed that such an α_k^n cannot be represented by a polynomial-sized circuit over $\{\neg,\vee,\wedge\}$ of depth at most $k-1$.

Consider the Boolean function α_4^n over n^6 variables. By construction, it can be represented by a disjunction of n conjunctions $\beta_1,\ldots,\beta_n$ of DNF formulae, where each β_i ($i \in 1,\ldots,n$) is the conjunction of n^2DNF$\gamma_{i,j}$ ($j \in 1,\ldots n^2$), each DNF$\gamma_{i,j}$ ($j \in 1,\ldots n^2$) consists of the disjunction of n^2 terms $\delta i,j,k$ ($k \in 1,\ldots n^2$), and finally each term $\delta i,j,k$ ($k \in 1,\ldots n^2$) consists of the conjunction of n negated variables $\neg x_{i,j,k,l}$ ($l \in 1,\ldots,n$) occurring only once in α_4^n. For each $i \in 1,\ldots,n$ and $j \in 1,\ldots n^2$, consider now the HORN formula $h_{i,j}$ such that

$$h_{i,j} = \left(\bigvee_{k=1}^{n^2} \neg y_{i,j,k}\right) \wedge \bigwedge_{k=1}^{n^2}\bigwedge_{l=1}^{n}(y_{i,j,k} \vee \neg x_{i,j,k,l}).$$

$h_{i,j}$ contains n^3+1 clauses of size at most n^2, hence the HORN formula $\bigwedge_{j=1}^{n^2} h_{i,j}$ contains n^5+n^2 clauses of size at most n^2. Let $Y = \bigcup_{i=1}^n(\bigcup_{j=1}^{n^2}(\bigcup_{k=1}^{n^2}\{y_{i,j,k}\}))$. By construction, the HORN[$\vee,\exists$] formula $\exists Y.(\bigvee_{i=1}^n(\bigwedge_{j=1}^{n^2} h_{i,j}))$ can be generated in time polynomial in n. From Sipser theorem, α_4^n has no polynomial-sized AC3 representation. It remains to show that α_4^n is equivalent to $\exists Y.(\bigvee_{i=1}^n(\bigwedge_{j=1}^{n^2} h_{i,j}))$. First of all, since existential quantifications "distribute" over disjunctions and since each $y_{i,j,k}$ ($i \in 1,\ldots,n$, $j \in 1,\ldots,n^2 k \in 1,\ldots,n^2$) does not occur in $h_{i',j'}$ ($i' \in 1,\ldots,n$, $j \in 1,\ldots,n^2$) unless $i'=i$ and $j'=j$, we have that $\exists Y.(\bigvee_{i=1}^n(\bigwedge_{j=1}^{n^2} h_{i,j}))$ is equivalent to $\bigvee_{i=1}^n(\bigwedge_{j=1}^{n^2}\exists\bigcup_{k=1}^{n^2}\{y_{i,j,k}\}.h_{i,j})$. Finally, by construction, for each $i \in 1,\ldots,n$ and $j \in 1,\ldots,n^2$, $\gamma_{i,j} = \bigvee_{k=1}^{n^2}\delta_{i,j,k}$ is the IP representation of $\exists\bigcup_{k=1}^{n^2}\{y_{i,j,k}\}.h_{i,j}$, hence it is equivalent to it. The replacement metatheorem for propositional logic concludes the proof. □

Proposition 11. *The results in Table 3 hold.*

	AFF[$\vee$]	renH[$\vee,\exists$]	K/H[$\vee,\exists$]	HORN[$\vee,\exists$]	KROM[$\vee$]
AFF[$\vee$]	$\sim_s$	$\not\leq_s$	$\not\leq_s$	$\not\leq_s$	$\not\leq_s$
renH[$\vee,\exists$]	$\not\leq_s$	$\sim_s$	$\leq_s$	$\leq_s$	$\leq_s$
K/H[$\vee,\exists$]	$\not\leq_s$	$\not\leq_s$	$\sim_s$	$\leq_s$	$\leq_s$
HORN[$\vee,\exists$]	$\not\leq_s$	$\not\leq_s$	$\not\leq_s$	$\sim_s$	$\not\leq_s$
KROM[$\vee$]	$\not\leq_s$	$\not\leq_s$	$\not\leq_s$	$\not\leq_s$	$\sim_s$

The relative succinctness of the full disjunctive closures of KROM, HORN, AFF, K/H, and renH.

Table 5
The relative succinctness of the full disjunctive closures of KROM, HORN, AFF, K/H, and renH.

	AFF[∨]	renH[∨, ∃]	K/H[∨, ∃]	HORN[∨, ∃]	KROM[∨]
AFF[∨]	$\mid\sim_s\mid$				
renH[∨, ∃]		$\mid\sim_s\mid$		$\mid\leq_s\mid$	
K/H[∨, ∃]			$\mid\sim_s\mid$	$\mid\leq_s\mid$	$\mid\leq_s\mid$
HORN[∨, ∃]				$\mid\sim_s\mid$	
KROM[∨]					$\mid\sim_s\mid$

Table 6
The relative succinctness of the full disjunctive closures of KROM, HORN, AFF, K/H, and renH.

	AFF[∨]	renH[∨, ∃]	K/H[∨, ∃]	HORN[∨, ∃]	KROM[∨]
AFF[∨]	$\sim_s$				
renH[∨, ∃]		$\sim_s$	$\mid\leq_s\mid$	$\leq_s$	$\mid\leq_s\mid$
K/H[∨, ∃]			$\sim_s$	$\leq_s$	$\leq_s$
HORN[∨, ∃]				$\sim_s$	
KROM[∨]					$\sim_s$

Table 7
The relative succinctness of the full disjunctive closures of KROM, HORN, AFF, K/H, and renH.

	AFF[∨]	renH[∨, ∃]	K/H[∨, ∃]	HORN[∨, ∃]	KROM[∨]
AFF[∨]	$\sim_s$				
renH[∨, ∃]		$\sim_s$	$\leq_s$	$\leq_s$	$\leq_s$
K/H[∨, ∃]			$\sim_s$	$\leq_s$	$\leq_s$
HORN[∨, ∃]		$\mid\not\leq_s\mid$	$\mid\not\leq_s\mid$	$\sim_s$	$\mid\not\leq_s\mid$
KROM[∨, ∃]					$\sim_s$

Proof. The proof is broken into six steps, where we prove some succinctness relationships between languages, and then apply transitivity of $\leq_s$ to possibly infer new relationships. Associated with each step of the proof is a table in which we mark all relationships proved at the step.

Table 5: From the obvious equalities and inclusions HORN[∨, ∃] ⊆ K/H[∨, ∃], HORN[∨, ∃] ⊆ renH[∨, ∃], KROM[∨] ⊆ K/H[∨, ∃], we get the results given in Table 5.

Table 6: Since K/H[∨, ∃] $\sim_p$ K/H[∃][∨] (cf. Proposition 1), every K/H[∨, ∃] formula can be associated in polynomial time with an equivalent disjunction $\bigvee_{i=1}^{n} \exists X_i.\beta_i$ of K/H[∃] formulae. Since KROM satisfies **CO**, we can easily determine in polynomial time which β_i are consistent. All the β_i ($i \in 1, \ldots, n$) which are inconsistent can be removed from the disjunction without questioning equivalence (if they are all inconsistent, the input formula is associated with $\bot$, which is a renH[∨, ∃] formula). In the remaining case, since every consistent KROM formula is a renH formula, the resulting disjunction is a renH[∨, ∃] formula equivalent to the input formula. Hence we get the results given in Table 6.

Table 7: Let us now show that HORN[∨, ∃] $\not\leq_s$ KROM[∨, ∃], HORN[∨, ∃] $\not\leq_s$ K/H[∨, ∃], and HORN[∨, ∃] $\not\leq_s$ renH[∨, ∃]. To do so, it is enough to prove that HORN[∨, ∃] $\not\leq_s$ KROM. Consider the KROM formula $\alpha_n = \bigwedge_{i=1}^{n}(x_i \vee y_i)$ for any n. Towards a contradiction, suppose that there exists in HORN[∨, ∃] a formula equivalent to α_n and whose size is polynomial in n; since HORN[∨, ∃] $\sim_p$ HORN[∃][∨] (cf. Proposition 1), there exists as well a HORN[∃][∨] formula $\beta = \bigvee_{i=1}^{m} \exists X_i.\beta_i$ equivalent to α_n and whose size is polynomial in n. Especially, m must remain polynomial in n. We also know that HORN[∃] $\sim_e$ HORN (cf. Proposition 7). Hence, if β exists, then there also exists a HORN[∨] formula $\gamma = \bigvee_{i=1}^{m} \gamma_i$ equivalent to α_n and with m polynomial in n. Note that the size of γ_i ($i \in 1, \ldots, m$) can be exponential in the size of β_i (this does not matter for the remaining part of the proof).

By construction, α_n has 2^n minimal models ω over $Var(\alpha_n)$, where for each $i \in 1, \ldots, n$, exactly one of the two variables x_i and y_i are set to 1 by ω. Consider now any pair ω, ω' of distinct minimal models of α_n; by construction, $and(\omega, \omega')$ maps each variable to 0, hence it is not a model of α_n. Thus, as a consequence of the characterization of HORN by closure of models, ω and ω' cannot be models of the same formula γ_i. Therefore, every HORN[∨] formula $\gamma = \bigvee_{i=1}^{m} \gamma_i$ equivalent to α_n must be such that $m \geq 2^n$. This shows that there is no HORN[∨, ∃] formula equivalent to α_n and whose size is polynomial in n. Thus, we get the results given in Table 7.

Table 8: Let us now show that KROM[∨] $\not\leq_s$ HORN[∨, ∃], KROM[∨] $\not\leq_s$ K/H[∨, ∃], and KROM[∨] $\not\leq_s$ renH[∨, ∃]. To do so, it is enough to prove that KROM[∨] $\not\leq_s$ HORN. Consider the HORN formula $\alpha_n = \bigwedge_{i=1}^{n}(\neg x_i \vee \neg y_i \vee \neg z_i)$ for any n. Towards a contradiction, suppose that there exists in KROM[∨] a formula $\gamma = \bigvee_{i=1}^{m} \gamma_i$ equivalent to α_n and whose size is polynomial in n; then m must remain polynomial in n.

Table 8
The relative succinctness of the full disjunctive closures of KROM, HORN, AFF, K/H, and renH.

	AFF[∨]	renH[∨, ∃]	K/H[∨, ∃]	HORN[∨, ∃]	KROM[∨]
AFF[∨]	$\sim_s$				
renH[∨, ∃]		$\sim_s$	$\leq_s$	$\leq_s$	$\leq_s$
K/H[∨, ∃]			$\sim_s$	$\leq_s$	$\leq_s$
HORN[∨, ∃]		$\not\leq_s$	$\not\leq_s$	$\sim_s$	$\not\leq_s$
KROM[∨]		$\not\leq_s$	$\not\leq_s$	$\not\leq_s$	$\sim_s$

Table 9
The relative succinctness of the full disjunctive closures of KROM, HORN, AFF, K/H, and renH.

	AFF[∨]	renH[∨, ∃]	K/H[∨, ∃]	HORN[∨, ∃]	KROM[∨]
AFF[∨]	$\sim_s$				
renH[∨, ∃]		$\sim_s$	$\leq_s$	$\leq_s$	$\leq_s$
K/H[∨, ∃]		$\not\leq_s$	$\sim_s$	$\leq_s$	$\leq_s$
HORN[∨, ∃]		$\not\leq_s$	$\not\leq_s$	$\sim_s$	$\not\leq_s$
KROM[∨]		$\not\leq_s$	$\not\leq_s$	$\not\leq_s$	$\sim_s$

By construction, α_n has 7^n models over $Var(\alpha_n)$. How many models of α_n can be models of the same γ_i ($i \in 1, \ldots, m$)? Let us consider any $\omega_1, \omega_2, \omega_3 \in Mod(\gamma_i)$, it *cannot* be the case that for any $i \in 1, \ldots, n$, we have $\omega_1(x_i) = 0$, $\omega_1(y_i) = 1$, $\omega_1(z_i) = 1$, $\omega_2(x_i) = 1$, $\omega_2(y_i) = 0$, $\omega_2(z_i) = 1$, $\omega_3(x_i) = 1$, $\omega_3(y_i) = 1$, $\omega_3(z_i) = 0$. Indeed, if this were the case, we would have $maj(\omega_1, \omega_2, \omega_3)(x_i) = maj(\omega_1, \omega_2, \omega_3)(y_i) = maj(\omega_1, \omega_2, \omega_3)(z_i) = 1$. If γ_i is a KROM formula, then $maj(\omega_1, \omega_2, \omega_3)$ should also be a model of γ_i. But $maj(\omega_1, \omega_2, \omega_3)$ is not a model of α_n. Thus, each γ_i cannot have more than 6^n models of α_n over $Var(\alpha_n)$. Subsequently, the pigeon/hole principle shows that at least $\lceil (\frac{7}{6})^n \rceil$KROM formulae γ_i are required to cover the models of α_n. The fact that $\lceil (\frac{7}{6})^n \rceil$ is exponential in the size of α_n concludes the proof. By transitivity of $\leq_s$, we get the results given in Table 8.

Table 9: We also have to show that K/H[∨, ∃] $\not\leq_s$ renH[∨, ∃]. To do so, it is enough to prove that K/H[∨, ∃] $\not\leq_s$ renH. Consider the renH formula $\alpha_n = \bigwedge_{i=1}^{n}(x_i \vee y_i \vee z_i)$ for any n ($Var(\alpha_n)$ is a possible Horn renaming for it, since if one replaces in α_n every literal from $L_{Var(\alpha_n)}$ by its complementary literal, one gets a HORN formula).

Towards a contradiction, suppose that there exists in K/H[∨, ∃] a formula equivalent to α_n and whose size is polynomial in n; since K/H[∨, ∃] $\sim_p$ K/H[∃][∨] (cf. Proposition 1), there exists as well a K/H[∃][∨] formula $\beta = \bigvee_{i=1}^{m} \exists X_i.\beta_i$ equivalent to α_n and whose size is polynomial in n. Especially, m must remain polynomial in n. We also know that K/H[∃] $\sim_e$ K/H (cf. Proposition 7). Hence, if β exists, then there also exists a K/H[∨] formula $\gamma = \bigvee_{i=1}^{m} \gamma_i$ equivalent to α_n and with m polynomial in n.

Let us now prove that if such a γ exists, then there also exists a KROM[∨] formula $\delta = \bigvee_{i=1}^{m} \delta_i$ equivalent to α_n and with m polynomial in n. Consider any K/H formula γ_i ($i \in 1, \ldots, m$) and suppose that it is a HORN formula. Then γ_i is equivalent to an implicant of α_n. This is obvious if γ_i is inconsistent. In the remaining case, every clause $x_i \vee y_i \vee z_i$ ($i \in 1, \ldots, n$) of α_n must be implied by a prime implicate of γ_i, which must be a Horn clause; hence this prime implicate must be equivalent to x_i, y_i or z_i. Accordingly, γ_i must be equivalent to a term, hence to a KROM formula.

It remains to show that α_n has no polyspace representation in KROM[∨]. The proof is similar to the one used for showing that KROM[∨, ∃] $\not\leq_s$ HORN (this is not surprising since α_n is a reverse Horn CNF formula). We get the results given in Table 9.

Table 10: Finally, we show that AFF[∨] is incomparable w.r.t. $\leq_s$ w.r.t. any of renH[∨, ∃], K/H[∨, ∃], HORN[∨, ∃] and KROM[∨]. We first show that renH[∨, ∃] $\not\leq_s$ AFF, which proves enough to conclude that renH[∨, ∃] $\not\leq_s$ AFF[∨], K/H[∨, ∃] $\not\leq_s$ AFF[∨], HORN[∨, ∃] $\not\leq_s$ AFF[∨], and KROM[∨] $\not\leq_s$ AFF[∨].

Let $\alpha_n = \bigwedge_{i=1}^{n}(x_i \oplus y_i \oplus z_i \oplus \top)$. By construction α_n is an AFF formula. Furthermore, the restriction of any model of α_n over any $\{x_i, y_i, z_i\}$ ($i \in 1, \ldots, n$) is of the form 000, 011, 101 or 110. Thus, α_n has 4^n models over $Var(\alpha_n)$. Let β be a renH[∨, ∃] formula equivalent to α_n. Since renH[∨, ∃] $\sim_p$ renH[∃][∨] (cf. Proposition 1), β is polynomially translatable into a formula $\bigvee_{i=1}^{m} \beta_i$ from renH[∃][∨]. Therefore, if β is a polynomial-sized representation of α_n, then $\bigvee_{i=1}^{m} \beta_i$ also is a polynomial-sized representation of α_n, which implies that m must not be exponential in n.

Since renH[∃] $\sim_e$ renH (cf. Proposition 7), each β_i ($i \in 1, \ldots, m$) can be translated into an equivalent renH formula γ_i (which size can be exponential in the size of β_i, but this does not matter). The point is that if β has a renH[∃][∨] representation as a disjunction of $m$renH[∃] formulae, then it also has a renH[∨] representation as a disjunction of $m$renH formulae.

Since each γ_i ($i \in 1, \ldots, m$) is a renH formula which entails α_n, from [75], there exists a model V_i of α_n such that V_i is a Horn renaming for γ_i, and $V_i(\gamma_i) \models V_i(\alpha_n)$. As explained above, the restriction of V_i over any $\{x_i, y_i, z_i\}$ ($i \in 1, \ldots, n$) is of the form 000, 011, 101 or 110. Since applying V_i leads to renaming an even number of variables in each set $\{x_i, y_i, z_i\}$ ($i \in 1, \ldots, n$), we necessarily have $V_i(x_i \oplus y_i \oplus z_i \oplus \top) \equiv x_i \oplus y_i \oplus z_i \oplus \top$, and subsequently $V_i(\alpha_n) \equiv \alpha_n$.

Table 10
The relative succinctness of the full disjunctive closures of KROM, HORN, AFF, K/H, and renH.

	AFF[∨]	renH[∨, ∃]	K/H[∨, ∃]	HORN[∨, ∃]	KROM[∨]
AFF[∨]	$\sim_s$	$\not\leq_s$	$\not\leq_s$	$\not\leq_s$	$\not\leq_s$
renH[∨, ∃]	$\not\leq_s$	$\sim_s$	$\leq_s$	$\leq_s$	$\leq_s$
K/H[∨, ∃]	$\not\leq_s$	$\not\leq_s$	$\sim_s$	$\leq_s$	$\leq_s$
HORN[∨, ∃]	$\not\leq_s$	$\not\leq_s$	$\not\leq_s$	$\sim_s$	$\not\leq_s$
KROM[∨]	$\not\leq_s$	$\not\leq_s$	$\not\leq_s$	$\not\leq_s$	$\sim_s$

Thus, we get that $\bigvee_{i=1}^{m} V_i(\gamma_i)$ is a HORN[∨] formula equivalent to α_n. At this stage, we have shown that if α_n has a polynomial-sized representation as a renH[∨, ∃] formula, then it must also have a HORN[∨] representation with a number of disjoints that is polynomial in n.

We are now going to prove that this is not the case, i.e., the number of disjoints in any HORN[∨] formula $\bigvee_{i=1}^{m} \delta_i$ equivalent to α_n actually is exponential in n. Consider the subset S of models ω of α_n over $Var(\alpha_n)$ such that for each $i \in 1, \ldots, n$ the restriction of ω over $\{x_i, y_i, z_i\}$ is of the form 011, 101 or 110. Every pair of distinct models ω and ω' from S is such that $and(\omega, \omega')$ is not a model of α_n; indeed, there must exist $i \in 1, \ldots, n$ such that the restrictions of ω and ω' over $\{x_i, y_i, z_i\}$ differ, and $and(\omega, \omega')$ is not a model of $x_i \oplus y_i \oplus z_i \oplus \top$ (its restriction over $\{x_i, y_i, z_i\}$ is of the form 001, 010 or 100). Thus, because of the closure property of HORN formulae, every pair of distinct models in S cannot be models of the same HORN formula δ_i. Since S contains 3^n models, the number m of disjoints in $\bigvee_{i=1}^{m} \delta_i$ is lower bounded by 3^n. This shows that α_n has no polynomial-sized representation as a renH[∨, ∃] formula.

Conversely, let us show that AFF[∨] is not at least as succinct as any of renH[∨, ∃], K/H[∨, ∃], HORN[∨, ∃] and KROM[∨]. Consider the formula $\alpha_n = \bigwedge_{i=1}^{n}(\neg x_i \vee \neg y_i)$ for any n. It is a KROM formula and a HORN formula. Hence, it is also a K/H formula and a renH formula. Since $Var(\alpha_n)$ contains $2n$ atoms, 4^n interpretations over $Var(\alpha_n)$ have to be considered. Among them, one can find 3^n models of α_n, only, since for each $i \in 1, \ldots, n$, there are only 3 truth assignments of x_i and y_i (over the four possible assignments of those two variables) which satisfy $\neg x_i \vee \neg y_i$. Now, there is no AFF formula β implying α_n and with strictly more than 2^n models (taken in the set of models of α_n since $\beta \models \alpha_n$ must hold). By *reductio ad absurdum*: if this were the case, then one could find $i \in 1, \ldots, n$ and $\omega_1, \omega_2, \omega_3 \in Mod(\alpha_n)$ such that $\omega_1(x_i) = 0$, $\omega_1(y_i) = 0$, $\omega_2(x_i) = 0$, $\omega_2(y_i) = 1$, $\omega_3(x_i) = 1$, $\omega_3(y_i) = 0$. If $\omega_1, \omega_2, \omega_3 \in Mod(\beta)$ and β is an AFF formula, then the affine closure property requires $\oplus(\omega_1, \omega_2, \omega_3)$ to be a model of β, hence a model of α_n. But $\oplus(\omega_1, \omega_2, \omega_3)$ falsifies $\neg x_i \vee \neg y_i$. Subsequently, from the pigeon/hole principle, every AFF[∨] formula equivalent to α_n must contain at least $\lceil(\frac{3}{2})^n\rceil$ AFF formulae as disjuncts. The fact that $\lceil(\frac{3}{2})^n\rceil$ is exponential in the size of α_n concludes the proof. □

Proposition 12. *The results in Table 4 hold.*

	AFF[∨]	renH[∨, ∃]	K/H[∨, ∃]	HORN[∨, ∃]	KROM[∨, ∃]
CNF	$\not\leq_s, \not\geq_s$	$\not\leq_s, \not\geq_s$	$\not\leq_s, \not\geq_s$	$\not\leq_s, \not\geq_s$	$\not\leq_s, \not\geq_s$
PI	$\not\leq_s, \not\geq_s$	$\not\leq_s, \not\geq_s$	$\not\leq_s, \not\geq_s$	$\not\leq_s, \not\geq_s$	$\not\leq_s, \not\geq_s$
DNNF_T	$\not\leq_s, \not\geq_s$	$\not\leq_s, \not\geq_s$	$\not\leq_s, \not\geq_s$	$\not\leq_s, \not\geq_s$	$\not\leq_s, \not\geq_s$
d-DNNF	$\not\leq_s^*, \not\geq_s$	$\not\leq_s^*, \not\geq_s$	$\not\leq_s^*, \not\geq_s$	$\not\leq_s^*, \not\geq_s$	$\not\leq_s^*, \not\geq_s$
DNF	$\not\leq_s, \geq_s$	$\not\leq_s, \geq_s$	$\not\leq_s, \geq_s$	$\not\leq_s, \geq_s$	$\not\leq_s, \geq_s$
IP	$\not\leq_s, \geq_s$	$\not\leq_s, \geq_s$	$\not\leq_s, \geq_s$	$\not\leq_s, \geq_s$	$\not\leq_s, \geq_s$
$\text{OBDD}_<$	$\not\leq_s, \not\geq_s$	$\not\leq_s, \not\geq_s$	$\not\leq_s, \not\geq_s$	$\not\leq_s, \not\geq_s$	$\not\leq_s, \not\geq_s$

Comparing w.r.t. succinctness the full disjunctive closures of KROM, HORN, K/H, renH, and AFF, with $\text{OBDD}_<$, IP, DNF, d-DNNF, DNNF_T, PI, and CNF. * means that the result holds unless the polynomial hierarchy collapses.

Proof. Again, the proof is broken in a number of steps, where we prove some succinctness relationships between languages, and then apply transitivity of $\leq_s$ to possibly infer new relationships. Associated with each step of the proof is a table in which we mark all relationships proved at the step.

Table 11: Let $\mathcal{L}$ be any language among AFF[∨], renH[∨], K/H[∨], HORN[∨], KROM[∨], and the corresponding existential closures renH[∨, ∃], K/H[∨, ∃], HORN[∨, ∃]. Since TERM ⊆ AFF, TERM ⊆ HORN, TERM ⊆ KROM, HORN ⊆ renH and KROM $\geq_p$ renH, we obviously have DNF $\geq_p \mathcal{L}$, hence we have DNF $\geq_s \mathcal{L}$. In [1], it is proven that PI $\geq_s$ CNF, DNF $\not\geq_s$ CNF, DNF $\not\geq_s$ $\text{OBDD}_<$, and IP $\geq_s$ DNF. By transitivity of $\leq_s$, we get the results given in Table 11.

Table 12: Consider the following consistent KROM formula $\alpha_n = \bigwedge_{i=1}^{n}(\neg x_i \vee \neg y_i)$; it is also a HORN formula, hence it belongs to the disjunction closure and to the full disjunctive closure of each language among KROM, HORN, K/H, and renH. α_n has 2^n essential prime implicants,[10] hence there is no polynomial-sized IP formula and no polynomial-sized DNF formula

[10] A prime implicant γ of a formula α is *essential* iff the disjunction of all prime implicants of α except γ (up to logical equivalence) is not equivalent to α.

Table 11
Comparing w.r.t. succinctness the full disjunctive closures of KROM, HORN, K/H, renH, and AFF, with other classes of propositional representations.

	AFF[∨]	renH[∨, ∃]	K/H[∨, ∃]	HORN[∨, ∃]	KROM[∨]
CNF	$\mid \not\geq_s \mid$	$\mid \not\geq_s \mid$	$\mid \not\geq_s \mid$	$\mid \not\geq_s \mid$	$\mid \not\geq_s \mid$
PI	$\mid \not\geq_s \mid$	$\mid \not\geq_s \mid$	$\mid \not\geq_s \mid$	$\mid \not\geq_s \mid$	$\mid \not\geq_s \mid$
DNNF_T					
d-DNNF					
DNF	$\mid \geq_s \mid$	$\mid \geq_s \mid$	$\mid \geq_s \mid$	$\mid \geq_s \mid$	$\mid \geq_s \mid$
IP	$\mid \geq_s \mid$	$\mid \geq_s \mid$	$\mid \geq_s \mid$	$\mid \geq_s \mid$	$\mid \geq_s \mid$
$\text{OBDD}_<$	$\mid \not\geq_s \mid$	$\mid \not\geq_s \mid$	$\mid \not\geq_s \mid$	$\mid \not\geq_s \mid$	$\mid \not\geq_s \mid$

Table 12
Comparing w.r.t. succinctness the full disjunctive closures of KROM, HORN, K/H, renH, and AFF, with other classes of propositional representations.

	AFF[∨]	renH[∨, ∃]	K/H[∨, ∃]	HORN[∨, ∃]	KROM[∨]
CNF	$\not\leq_s$	$\not\leq_s$	$\not\leq_s$	$\not\leq_s$	$\not\leq_s$
PI	$\not\leq_s$	$\not\leq_s$	$\not\leq_s$	$\not\leq_s$	$\not\leq_s$
DNNF_T					
d-DNNF					
DNF	$\mid \not\leq_s \mid, \geq_s$	$\mid \not\leq_s \mid, \geq_s$	$\mid \not\leq_s \mid, \geq_s$	$\mid \not\leq_s \mid, \geq_s$	$\mid \not\leq_s \mid, \geq_s$
IP	$\mid \not\leq_s \mid, \geq_s$	$\mid \not\leq_s \mid, \geq_s$	$\mid \not\leq_s \mid, \geq_s$	$\mid \not\leq_s \mid, \geq_s$	$\mid \not\leq_s \mid, \geq_s$
$\text{OBDD}_<$	$\not\leq_s$	$\not\leq_s$	$\not\leq_s$	$\not\leq_s$	$\not\leq_s$

Table 13
Comparing w.r.t. succinctness the full disjunctive closures of KROM, HORN, K/H, renH, and AFF, with other classes of propositional representations.

	AFF[∨]	renH[∨, ∃]	K/H[∨, ∃]	HORN[∨, ∃]	KROM[∨]
CNF	$\not\leq_s, \mid \not\geq_s \mid$	$\not\leq_s, \mid \not\geq_s \mid$	$\not\leq_s, \mid \not\geq_s \mid$	$\not\leq_s, \mid \not\geq_s \mid$	$\not\leq_s, \mid \not\geq_s \mid$
PI	$\not\leq_s, \mid \not\geq_s \mid$	$\not\leq_s, \mid \not\geq_s \mid$	$\not\leq_s, \mid \not\geq_s \mid$	$\not\leq_s, \mid \not\geq_s \mid$	$\not\leq_s, \mid \not\geq_s \mid$
DNNF_T	$\mid \not\geq_s \mid$	$\mid \not\geq_s \mid$	$\mid \not\geq_s \mid$	$\mid \not\geq_s \mid$	$\mid \not\geq_s \mid$
d-DNNF	$\mid \not\geq_s \mid$	$\mid \not\geq_s \mid$	$\mid \not\geq_s \mid$	$\mid \not\geq_s \mid$	$\mid \not\geq_s \mid$
DNF	$\not\leq_s, \geq_s$	$\not\leq_s, \geq_s$	$\not\leq_s, \geq_s$	$\not\leq_s, \geq_s$	$\not\leq_s, \geq_s$
IP	$\not\leq_s, \geq_s$	$\not\leq_s, \geq_s$	$\not\leq_s, \geq_s$	$\not\leq_s, \geq_s$	$\not\leq_s, \geq_s$
$\text{OBDD}_<$	$\not\leq_s, \mid \not\geq_s \mid$	$\not\leq_s, \mid \not\geq_s \mid$	$\not\leq_s, \mid \not\geq_s \mid$	$\not\leq_s, \mid \not\geq_s \mid$	$\not\leq_s, \mid \not\geq_s \mid$

equivalent to it. Similarly, the AFF formula $\beta_n = \bigoplus_{i=1}^n x_i$ (which is also an AFF[∨] formula) has 2^{n-1} essential prime implicants, hence there is no polynomial-sized IP formula and no polynomial-sized DNF formula equivalent to it. We get the results given in Table 12.

Table 13: In the proof of Proposition 11, we have shown that the AFF formula $\alpha_n = \bigwedge_{i=1}^n (x_i \oplus y_i \oplus z_i \oplus \top)$ has no polynomially-sized renH[∨, ∃] representation. The point is that α_n has a polynomially-sized PI representation (consisting in $4n$ clauses: $\neg x_i \vee \neg y_i \vee \neg z_i$, $\neg x_i \vee y_i \vee z_i$, $x_i \vee \neg y_i \vee z_i$, $x_i \vee y_i \vee \neg z_i$ for each $i \in 1, \ldots, n$), and a polynomially-sized $\text{OBDD}_<$ representation for every ordering $<$ which is such that x_i, y_i, z_i ($i \in 1, \ldots, n$) are successive elements. Indeed, for each $i \in 1, \ldots, n$, one can generate in constant time an $\text{OBDD}_<$ representation equivalent to each $x_i \oplus y_i \oplus z_i \oplus \top$ and then, starting with the $\text{OBDD}_<$ representation of $x_1 \oplus y_1 \oplus z_1 \oplus \top$, in an iterative way, replace the $\top$ sink of the current $\text{OBDD}_<$ representation by the root of the next $\text{OBDD}_<$ representation.

Furthermore, in the proof of Proposition 11, we proved that the formula $\alpha_n = \bigwedge_{i=1}^n (\neg x_i \vee \neg y_i)$ (for any n) does not have a polynomial-size AFF[∨] representation. The point is that α_n is a PI formula, and it also has a polynomially-sized $\text{OBDD}_<$ representation for every ordering $<$ which is such that x_i, y_i ($i \in 1, \ldots, n$) are successive elements. Indeed, for each $i \in 1, \ldots, n$, one can generate in constant time an $\text{OBDD}_<$ representation equivalent to each $\neg x_i \vee \neg y_i$ and then, starting with the $\text{OBDD}_<$ representation of $\neg x_1 \vee \neg y_1$, in an iterative way, replace the $\top$ sink of the current $\text{OBDD}_<$ representation by the root of the next $\text{OBDD}_<$ representation.

Given that $\text{PI} \geq_p \text{CNF}$, $\text{OBDD}_< \geq_p$ d-DNNF, $\text{OBDD}_< \geq_p \text{DNNF}_\text{T}$, and the succinctness relationships given in Proposition 11, by transitivity of $\leq_s$, we get the results given in Table 13.

Table 14: As to DNNF_T, it is enough to show that the family of circular bit shift functions cbs_m have polynomially-sized representations in KROM[∨], HORN[∨], K/H[∨], and AFF[∨]. Indeed, it has been proven that such functions do not have polynomially-sized SDNNF representations, where SDNNF is the union of DNNF_T for all vtrees T [20].

For any positive integer m, consider the following Boolean function over $2^{m+1} + m$ variables $cbs_m(x_0, \ldots, x_{2^m-1}, y_0, \ldots, y_{2^m-1}, i_0, \ldots, i_{m-1})$ which is the semantics of the formula

$$\alpha_m = \bigvee_{b_0,\ldots,b_{m-1} \in \{0,1\}} \left(\bigwedge_{j=0}^{m-1} i_j^{b_j} \wedge \bigwedge_{j=0}^{2^m-1} x_j \Leftrightarrow y_{(j+num(b_0,\ldots,b_{m-1}))mod2^m} \right),$$

Table 14
Comparing w.r.t. succinctness the full disjunctive closures of KROM, HORN, K/H, renH, and AFF, with other classes of propositional representations.

	AFF[∨]	renH[∨, ∃]	K/H[∨, ∃]	HORN[∨, ∃]	KROM[∨]
CNF	$\not\leq_s, \not\geq_s$	$\not\leq_s, \not\geq_s$	$\not\leq_s, \not\geq_s$	$\not\leq_s, \not\geq_s$	$\not\leq_s, \not\geq_s$
PI	$\not\leq_s, \not\geq_s$	$\not\leq_s, \not\geq_s$	$\not\leq_s, \not\geq_s$	$\not\leq_s, \not\geq_s$	$\not\leq_s, \not\geq_s$
DNNF_T	$\mathbf{I}\not\leq_s\mathbf{I}, \not\geq_s$	$\mathbf{I}\not\leq_s\mathbf{I}, \not\geq_s$	$\mathbf{I}\not\leq_s\mathbf{I}, \not\geq_s$	$\mathbf{I}\not\leq_s\mathbf{I}, \not\geq_s$	$\mathbf{I}\not\leq_s\mathbf{I}, \not\geq_s$
d-DNNF	$\mathbf{I}\not\leq^*_s\mathbf{I}, \not\geq_s$	$\mathbf{I}\not\leq^*_s\mathbf{I}, \not\geq_s$	$\mathbf{I}\not\leq^*_s\mathbf{I}, \not\geq_s$	$\mathbf{I}\not\leq^*_s\mathbf{I}, \not\geq_s$	$\mathbf{I}\not\leq^*_s\mathbf{I}, \not\geq_s$
DNF	$\not\leq_s, \geq_s$	$\not\leq_s, \geq_s$	$\not\leq_s, \geq_s$	$\not\leq_s, \geq_s$	$\not\leq_s, \geq_s$
IP	$\not\leq_s, \geq_s$	$\not\leq_s, \geq_s$	$\not\leq_s, \geq_s$	$\not\leq_s, \geq_s$	$\not\leq_s, \geq_s$
$\text{OBDD}_<$	$\not\leq_s, \not\geq_s$	$\not\leq_s, \not\geq_s$	$\not\leq_s, \not\geq_s$	$\not\leq_s, \not\geq_s$	$\not\leq_s, \not\geq_s$

whose size is linear in the number of variables of cbs_m. In this formula, $i_j^{b_j}$ denotes the literal i_j when $b_j = 0$ and the literal $\neg i_j$ when $b_j = 1$; num is the mapping from $\{0,1\}^m$ to the set of natural numbers which gives the integer represented by the binary string $b_0 \ldots b_{m-1}$. Thus, the variables $i_0, \ldots, i_{m-1}$ make precise how the bits of the binary string $y_0 \ldots y_{2^m-1}$ must be (circularly) shifted, and $cbs_m(x_0, \ldots, x_{2^m-1}, y_0, \ldots, y_{2^m-1}, i_0, \ldots, i_{m-1}) = 1$ exactly when the variables $x_0, \ldots, x_{2^m-1}$ and the shifted variables $y_0, \ldots, y_{2^m-1}$ are pairwise equal.

For each $b_0, \ldots, b_{m-1} \in \{0,1\}$, the formula

$$\beta_{b_0,\ldots,b_{m-1}} = \bigwedge_{j=0}^{m-1} i_j^{b_j} \wedge \bigwedge_{j=0}^{2^m-1} x_j \Leftrightarrow y_{(j+num(b_0,\ldots,b_{m-1}))mod2^m}$$

is equivalent to the KROM formula

$$\gamma_{b_0,\ldots,b_{m-1}} = \bigwedge_{j=0}^{m-1} i_j^{b_j} \wedge \bigwedge_{j=0}^{2^m-1} (\neg x_j \vee y_{(j+num(b_0,\ldots,b_{m-1}))mod2^m}) \wedge \bigwedge_{j=0}^{2^m-1} (x_j \vee \neg y_{(j+num(b_0,\ldots,b_{m-1}))mod2^m}).$$

Clearly enough, $\gamma_{b_0,\ldots,b_{m-1}}$ also is a HORN formula, hence it is a K/H formula and a renH formula. Similarly, $\beta_{b_0,\ldots,b_{m-1}}$ is also equivalent to the AFF formula

$$\delta b_0, \ldots, b_{m-1} = \bigwedge_{j=0}^{m-1} lit(i_j, b_j) \wedge \bigwedge_{j=0}^{2^m-1} x_j \oplus y_{(j+num(b_0,\ldots,b_{m-1}))mod2^m} \oplus \top$$

where $lit(i_j, b_j) = i_j$ when $b_j = 0$ and $lit(i_j, b_j) = i_j \oplus \top$ when $b_j = 1$. Both $\gamma_{b_0,\ldots,b_{m-1}}$ and $\delta_{b_0,\ldots,b_{m-1}}$ can be computed in time linear in the size of $\beta_{b_0,\ldots,b_{m-1}}$, hence linear in the number of variables of cbs_m.

As a consequence, $\bigvee_{b_0,\ldots,b_{m-1}\in\{0,1\}} \gamma_{b_0,\ldots,b_{m-1}}$ is a KROM[∨] (and a HORN[∨], a K/H[∨], a renH[∨]) formula equivalent to α_m, and $\bigvee_{b_0,\ldots,b_{m-1}\in\{0,1\}} \delta_{b_0,\ldots,b_{m-1}}$ is an AFF[∨] formula equivalent to α_m. The fact that the size of any of

$$\bigvee_{b_0,\ldots,b_{m-1}\in\{0,1\}} \gamma_{b_0,\ldots,b_{m-1}}$$

and

$$\bigvee_{b_0,\ldots,b_{m-1}\in\{0,1\}} \delta_{b_0,\ldots,b_{m-1}}$$

is linear in the number of variables of cbs_m completes the proof.

As to d-DNNF, the result comes easily from the fact that d-DNNF is not at least as succinct as DNF, unless the polynomial hierarchy collapses [1], plus the fact that DNF is polynomially translatable into the disjunction closure and into the full disjunctive closure of each of KROM, HORN, K/H, and renH.

We finally get the results given in Table 14. □

References

[1] A. Darwiche, P. Marquis, A knowledge compilation map, J. Artif. Intell. Res. 17 (2002) 229–264.
[2] H. Fargier, P. Marquis, Extending the knowledge compilation map: Krom, Horn, affine and beyond, in: Proc. of AAAI'08, 2008, pp. 442–447.
[3] P. Marquis, Existential closures for knowledge compilation, in: Proc. of IJCAI'11, 2011, pp. 996–1001.
[4] H. Fargier, P. Marquis, Extending the knowledge compilation map: closure principles, in: Proc. of ECAI'08, 2008, pp. 50–54.
[5] M. Cadoli, F. Donini, A survey on knowledge compilation, AI Commun. 10 (3–4) (1998) 137–150.
[6] Y. Boufkhad, E. Grégoire, P. Marquis, B. Mazure, L. Saïs, Tractable cover compilations, in: Proc. of IJCAI'97, 1997, pp. 122–127.
[7] B. Selman, H. Kautz, Knowledge compilation and theory approximation, J. ACM 43 (1996) 193–224.
[8] R. Schrag, Compilation for critically constrained knowledge bases, in: Proc. of AAAI'96, 1996, pp. 510–515.
[9] P. Marquis, Knowledge compilation using theory prime implicates, in: Proc. of IJCAI'95, 1995, pp. 837–843.

[10] A. del Val, Tractable databases: how to make propositional unit resolution complete through compilation, in: Proc. of KR'94, 1994, pp. 551–561.
[11] R. Dechter, I. Rish, Directional resolution: the Davis–Putnam procedure, revisited, in: Proc. of KR'94, 1994, pp. 134–145.
[12] A. Darwiche, Decomposable negation normal form, J. ACM 48 (4) (2001) 608–647.
[13] G. Gogic, H. Kautz, C. Papadimitriou, B. Selman, The comparative linguistics of knowledge representation, in: Proc. of IJCAI'95, 1995, pp. 862–869.
[14] M. Wachter, R. Haenni, Propositional DAGs: a new graph-based language for representing Boolean functions, in: Proc. of KR'06, 2006, pp. 277–285.
[15] H. Fargier, P. Marquis, On the use of partially ordered decision graphs in knowledge compilation and quantified Boolean formulae, in: Proc. of AAAI'06, 2006, pp. 42–47.
[16] S. Subbarayan, L. Bordeaux, Y. Hamadi, Knowledge compilation properties of tree-of-BDDs, in: Proc. of AAAI'07, 2007, pp. 502–507.
[17] R. Mateescu, R. Dechter, R. Marinescu, AND/OR multi-valued decision diagrams (AOMDDs) for graphical models, J. Artif. Intell. Res. 33 (2008) 465–519.
[18] K. Pipatsrisawat, A. Darwiche, New compilation languages based on structured decomposability, in: Proc. of AAAI'08, 2008, pp. 517–522.
[19] H. Fargier, P. Marquis, Knowledge compilation properties of trees-of-BDDs, revisited, in: Proc. of IJCAI'09, 2009, pp. 772–777.
[20] K. Pipatsrisawat, A. Darwiche, A lower bound on the size of decomposable negation normal form, in: Proc. of AAAI'10, 2010.
[21] A. Darwiche, SDD: a new canonical representation of propositional knowledge bases, in: Proc. of IJCAI'11, 2011, pp. 819–826.
[22] L. Bordeaux, J. Marques-Silva, Knowledge compilation with empowerment, in: Proc. of SOFSEM'12, 2012, pp. 612–624.
[23] M. Krom, The decision problem for formulas in prenex conjunctive normal form with binary disjunctions, J. Symb. Log. 35 (1970) 210–216.
[24] A. Horn, On sentences which are true of direct unions of algebras, J. Symb. Log. 16 (1951) 14–21.
[25] T.J. Schaefer, The complexity of satisfiability problems, in: Proc. of STOC'78, 1978, pp. 216–226.
[26] H. Lewis, Renaming a set of clauses as a Horn set, J. Assoc. Comput. Mach. 25 (1978) 134–135.
[27] H. Kleine-Büning, X. Zhao, U. Bubeck, Transformations into normal forms for quantified circuits, in: Proc. of SAT'11, 2011, pp. 245–258.
[28] J.J. Hébrard, A linear algorithm for renaming a set of clauses as a Horn set, Theor. Comput. Sci. 124 (2) (1994) 343–350.
[29] A. del Val, On 2-sat and renamable Horn, in: Proc. of AAAI'00, 2000, pp. 279–284.
[30] E. Post, The two-valued iterative systems of mathematical logic, Ann. Math. Stud. 5 (1941) 1–122.
[31] J. McKinsey, The decision problem for some classes of sentences without quantifiers, J. Symb. Log. 8 (1943) 61–77.
[32] R. Dechter, J. Pearl, Structure identification in relational data, Artif. Intell. 58 (1992) 237–270.
[33] A. Darwiche, Model-based diagnosis using structured system descriptions, J. Artif. Intell. Res. 8 (1998) 165–222.
[34] A. Herzig, J. Lang, P. Marquis, T. Polacsek, Updates, actions, and planning, in: Proc. of IJCAI'01, 2001, pp. 119–124.
[35] J. Lang, P. Marquis, Resolving inconsistencies by variable forgetting, in: Proc. of KR'02, 2002, pp. 239–250.
[36] J. Lang, P. Marquis, Reasoning under inconsistency: a forgetting-based approach, Artif. Intell. 174 (12–13) (2010) 799–823.
[37] H. Fargier, J. Lang, P. Marquis, Propositional logic and one-stage decision making, in: Proc. of KR'00, 2000, pp. 445–456.
[38] G. Cantor, Beiträge zur Begründung der Transfiniten Mengenlehre, Math. Ann. 49 (1897) 207–246.
[39] J. Hastad, Almost optimal lower bounds for small depth circuits, in: Proc. of STOC'86, 1986, pp. 6–20.
[40] G. Tseitin, On the complexity of derivation in propositional calculus, in: Structures in Constructive Mathematics and Mathematical Logic, Steklov Mathematical Institute, 1968, pp. 115–125.
[41] J. Huang, A. Darwiche, On compiling system models for faster and more scalable diagnosis, in: Proc. of AAAI'05, 2005, pp. 300–306.
[42] P. Torasso, G. Torta, Model-based diagnosis through OBDD compilation: a complexity analysis, in: Reasoning, Action and Interaction in AI Theories and Systems, 2006, pp. 287–305.
[43] J.-M. Astesana, L. Cosserat, H. Fargier, Constraint-based vehicle configuration: a case study, in: ICTAI, vol. 1, 2010, pp. 68–75.
[44] A. Felfernig, G. Friedrich, D. Jannach, M. Zanker, Developing constraint-based recommenders, in: Recommender Systems Handbook, 2011, pp. 187–215.
[45] C. Zengler, W. Küchlin, Boolean quantifier elimination for automotive configuration – a case study, in: Proc. of FMICS'13, 2013, pp. 48–62.
[46] S. Khurshid, D. Marinov, I. Shlyakhter, D. Jackson, A case for efficient solution enumeration, in: Proc. of SAT'03, 2003, pp. 272–286.
[47] O. Grumberg, A. Schuster, A. Yadgar, Memory efficient all-solutions SAT solver and its application for reachability analysis, in: Proc. of FMCAD'04, 2004, pp. 275–289.
[48] M. Gebser, B. Kaufmann, T. Schaub, Solution enumeration for projected Boolean search problems, in: Proc. of CP-AI-OR'09, 2009, pp. 71–86.
[49] S.K. Lahiri, R.E. Bryant, B. Cook, A symbolic approach to predicate abstraction, in: Proc. of CAV'03, 2003, pp. 141–153.
[50] P. Chauhan, E.M. Clarke, D. Kroening, A SAT-based algorithm for reparameterization in symbolic simulation, in: Proc. of DAC'04, 2004, pp. 524–529.
[51] A. Gupta, Z. Yang, P. Ashar, A. Gupta, SAT-based image computation with application in reachability analysis, in: Proc. of FMCAD'00, 2000, pp. 354–371.
[52] H.-J. Kang, I.-C. Park, SAT-based unbounded symbolic model checking, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 24 (2) (2005) 129–140.
[53] K.L. McMillan, Applying SAT methods in unbounded symbolic model checking, in: Proc. of CAV'02, 2002, pp. 250–264.
[54] M. Gelfond, H. Przymusinska, T. Przymusinski, On the relationship between circumscription and negation as failure, Artif. Intell. 38 (1989) 49–73.
[55] A. Yahya, L. Henschen, Deduction in non-Horn databases, J. Autom. Reason. 1 (1985) 141–160.
[56] J. Minker, On indefinite databases and the closed world assumption, in: Proc. of CADE'82, 1982, pp. 292–308.
[57] M. Gelfond, H. Przymusinska, Negation as failure: careful closure procedure, Artif. Intell. 30 (1986) 273–287.
[58] S. Coste-Marquis, P. Marquis, Knowledge compilation for closed world reasoning and circumscription, J. Log. Comput. 11 (4) (2001) 579–607.
[59] N. Nishimura, P. Ragde, S. Szeider, Detecting backdoor sets with respect to Horn and binary clauses, in: Proc. of SAT'04, 2004.
[60] M. Samer, S. Szeider, Backdoor trees, in: Proc. of AAAI'08, 2008, pp. 363–368.
[61] S. Gaspers, S. Szeider, Backdoors to satisfaction, in: The Multivariate Algorithmic Revolution and Beyond, 2012, pp. 287–317.
[62] E.M. Clarke, O. Grumberg, D. Peled, Model Checking, MIT Press, 2001.
[63] S. Kleene, Mathematical Logic, John Wiley, 1967.
[64] R. Tarjan, Depth first search and linear graph algorithms, SIAM J. Comput. 1 (1972) 146–160.
[65] B. Aspvall, M. Plass, R. Tarjan, A linear-time algorithm for testing the truth of certain quantified Boolean formulas, Inf. Process. Lett. 8 (1979) 121–123, Erratum: Inf. Process. Lett. 14 (4) (1982) 195.
[66] B. Aspvall, Recognizing disguised NR(1) instances of the satisfiability problem, J. Algorithms 1 (1) (1980) 97–103.
[67] W. Dowling, J. Gallier, Linear time algorithms for testing the satisfiability of propositional Horn formulae, J. Log. Program. 1 (3) (1984) 267–284.
[68] M. Karpinski, H.K. Büning, P.H. Schmitt, On the computational complexity of quantified Horn clauses, in: Proc. of CSL'87, 1987, pp. 129–137.
[69] N. Creignou, M. Hermann, Complexity of generalized satisfiability counting problems, Inf. Comput. 125 (1) (1996) 1–12.
[70] D. Roth, On the hardness of approximate reasoning, Artif. Intell. 82 (1–2) (1996) 273–302.
[71] P. Marquis, Consequence finding algorithms, in: Handbook on Defeasible Reasoning and Uncertainty Management Systems, in: Algorithms for Defeasible and Uncertain Reasoning, vol. 5, Kluwer Academic Publisher, 2000, pp. 41–145.
[72] B. Zanuttini, New polynomial classes for logic-based abduction, J. Artif. Intell. Res. 19 (2003) 1–10.
[73] U. Bubeck, H.K. Büning, Models and quantifier elimination for quantified Horn formulas, Discrete Appl. Math. 156 (10) (2008) 1606–1622.
[74] M. Sipser, Borel sets and circuit complexity, in: Proc. of STOC'83, 1983, pp. 61–69.
[75] Y. Boufkhad, Algorithms for propositional KB approximation, in: Proc. of AAAI'98, Madison (WI), 1998, pp. 280–285.