Existential Closures for Knowledge Compilation[*]

Pierre Marquis
CRIL-CNRS, Université d'Artois,
Lens, France
marquis@cril.univ-artois.fr

Abstract

We study the existential closures of several propositional languages $\mathcal{L}$ considered recently as target languages for knowledge compilation (KC), namely the incomplete fragments KROM−C, HORN−C, K/H−C, renH−C, AFF, and the corresponding disjunction closures KROM−C[$\vee$], HORN−C[$\vee$], K/H−C[$\vee$], renH−C[$\vee$], and AFF[$\vee$]. We analyze the queries, transformations, expressiveness and succinctness of the resulting languages $\mathcal{L}[\exists]$ in order to locate them in the KC map. As a by-product, we also address several issues concerning disjunction closures that were left open so far. From our investigation, the language $\mathtt{HORN}-\mathtt{C}[\vee, \exists]$ (where disjunctions and existential quantifications can be applied to Horn CNF formulae) appears as an interesting target language for the KC purpose, challenging the influential DNNF languages.

1 Introduction

Knowledge compilation (KC) is concerned with pre-processing for improving the efficiency of computational tasks (see among others [Darwiche, 2001; Cadoli and Donini, 1998; Selman and Kautz, 1996; Schrag, 1996; del Val, 1994]). An important issue within this research area is the choice of a target language into which some pieces of data are to be translated during the off-line, compilation phase [Gogic *et al.*, 1995; Darwiche and Marquis, 2002]. In [Darwiche and Marquis, 2002], the authors argue that such a choice must be based both on the set of queries and transformations which can be achieved in polynomial time once the pre-processed pieces of data are represented in the target language, as well as the succinctness of the language (i.e., its ability to represent data using few space.) The basic queries considered in [Darwiche and Marquis, 2002] include tests for consistency, validity, implicates (clausal entailment), implicants, equivalence, sentential entailment, counting and enumerating theory models (**CO**, **VA**, **CE**, **IM**, **EQ**, **SE**, **CT**, **ME**.) The basic transformations are conditioning (**CD**), closures under the connectives ($\wedge$**C**, $\wedge$**BC**, $\vee$**C**, $\vee$**BC**, $\neg$**C**), and forgetting (**FO**, **SFO**.) The KC map given in [Darwiche and Marquis, 2002] is an evaluation of a dozen of significant propositional languages w.r.t. several criteria: the succinctness of the language and the set of queries and transformations it supports in polynomial time. Such a map can be used as a guide for targeting the "right language" given the requirements imposed by the application under consideration.

The KC map provided in [Darwiche and Marquis, 2002] has been extended to incorporate further propositional languages (also referred to as "fragments"), queries and transformations, see among others [Wachter and Haenni, 2006], [Fargier and Marquis, 2006], [Subbarayan *et al.*, 2007], [Fargier and Marquis, 2008b], [Fargier and Marquis, 2008a], [Fargier and Marquis, 2009]. In [Fargier and Marquis, 2008b; 2008a] an approach to define new target languages for KC has been pointed out. It consists in applying closure principles to previous languages. [Fargier and Marquis, 2008a] consider two disjunctive closure principles: disjunction ($\vee$) and existential closure ($\exists$.) Intuitively, the disjunction principle when applied to a propositional language $\mathcal{L}$ leads to a language $\mathcal{L}[\vee]$ which qualifies disjunctions of formulae from $\mathcal{L}$, while existential closure applied to $\mathcal{L}$ leads to a language $\mathcal{L}[\exists]$ which qualifies existentially quantified formulae from $\mathcal{L}$. Whatever $\mathcal{L}$, $\mathcal{L}[\vee]$ satisfies polytime closure under $\vee$ and $\mathcal{L}[\exists]$ satisfies polytime forgetting. Applying any/both of those two principles to $\mathcal{L}$ may lead to new fragments, which can prove strictly more succinct than $\mathcal{L}$. Thus, [Fargier and Marquis, 2008a] locate on the KC map all languages obtained by applying those closure principles to some complete languages, i.e., languages $\mathcal{L}$ for which every propositional formula has an equivalent in $\mathcal{L}$. Thus, the disjunctive closures of the languages $\mathtt{OBDD}_<$ (ordered binary decision diagrams), DNF (disjunctive normal forms), DNNF (decomposable negation normal forms), CNF (conjunctive normal forms), PI (prime implicates), IP (prime implicants), MODS (models), considered in [Darwiche and Marquis, 2002], have been studied in [Fargier and Marquis, 2008a]. On the other hand, [Fargier and Marquis, 2008b] consider the disjunction closures KROM−C[$\vee$], HORN−C[$\vee$], K/H−C[$\vee$], renH−C[$\vee$], and AFF[$\vee$], composed respectively of disjunctions of Krom CNF formulae, disjunctions of Horn CNF formulae, disjunctions of Krom or Horn CNF formulae, disjunctions of renamable Horn CNF formulae, and disjunctions of affine formulae.

[*]This research was supported by the project TUPLES-ANR-10-BLAN-0210 of the French National Agency for Research - Agence Nationale de la Recherche

Each of these languages is complete, unlike the underlying languages `KROM-C` (also known as the bijunctive fragment) [Krom, 1970], `HORN-C` [Horn, 1951], `AFF` (also known as the biconditional fragment) [Schaefer, 1978], `K/H-C` (Krom or Horn CNF formulae) and `renH-C`.

In the following, we complete the results reported in [Fargier and Marquis, 2008b; 2008a] by focusing on the existential closures of the ten languages `KROM-C`, `HORN-C`, `K/H-C`, `renH-C`, `AFF`, `KROM-C`[$\vee$], `HORN-C`[$\vee$], `K/H-C`[$\vee$], `renH-C`[$\vee$], and `AFF`[$\vee$]. We evaluate each of them along the lines considered in [Darwiche and Marquis, 2002]. The contribution of the paper is mainly as follows:

- For each existential closure of `KROM-C`, `HORN-C`, `K/H-C`, `renH-C`, `AFF`, `KROM-C`[$\vee$], `HORN-C`[$\vee$], `K/H-C`[$\vee$], `renH-C`[$\vee$], and `AFF`[$\vee$], we identify the queries and transformations which are feasible in polynomial time, and those which are not (possibly under some standard assumptions of complexity theory.)
- We demonstrate that the existential closure of each language $\mathcal{L}$ among the ten languages above is just as expressive as $\mathcal{L}$. As to succinctness, we prove that $\mathcal{L}[\exists]$ is strictly more succinct than $\mathcal{L}$, except for $\mathcal{L}$ = `KROM-C` and $\mathcal{L}$ = `AFF` since for those two languages, $\mathcal{L}[\exists]$ and $\mathcal{L}$ are polynomially equivalent.
- We complete the results about disjunction closures provided in [Fargier and Marquis, 2008b]; especially, we show that **FO** is not satisfied by any of `HORN-C`[$\vee$], `K/H-C`[$\vee$], `renH-C`[$\vee$].
- We show that neither `d-DNNF` nor `SDNNF` is strictly more succinct than any disjunction closure of `KROM-C`, `HORN-C`, `K/H-C`, `renH-C`, `AFF`; it shows such disjunction closures (and the existential closures of them) as interesting alternatives to DNNF languages.

The rest of the paper is organized as follows: in Section 2, the queries and transformations considered in the KC map, as well as the fundamental notions of expressiveness and succinctness, are recalled. The notions of disjunctive, disjunction and existential closure are also presented. In Section 3, our new results are presented. Section 4 discusses the results and provides some perspectives. An extended version of the paper (including proofs) is available at `http://www.cril.fr/~marquis/ijcai11.pdf`

2 The KC Map and Disjunctive Closures

In this paper, we consider subsets of the propositional language `QDAG` of quantified propositional DAGs. `QDAG` is given by:

Definition 1 (`QDAG`) *Let PS be a denumerable set of propositional variables.* `QDAG` *is the set of all finite, single-rooted DAGs α where:*

- *each leaf node of α is labeled by a literal l over PS, or by a Boolean constant $\top$ (always true) or $\bot$ (always false) ;*
- *each internal node of α is labeled by a connective $c \in \{\wedge, \vee, \neg, \oplus\}$ and has as many children as required by c, or is labeled by $\exists x$ (where $x \in PS$) and has a single child.*

All the propositional languages considered so far as target languages to KC are subsets of `QDAG`, and typically of `DAG`, the subset of `QDAG` where no node labeled by a quantification is allowed. Especially, the fragments considered in [Wachter and Haenni, 2006] are included in the subset of `DAG` where c is among $\{\wedge, \vee, \neg\}$, and the fragments considered in [Darwiche and Marquis, 2002] (especially `OBDD`$_<$, `DNF`, `DNNF`, `CNF`, `PI`, `IP`, `MODS`) are subsets of `DAG-NNF` (the subset of the latter when $\neg$ is disallowed.)

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. $\overline{l}$ is the complementary literal of literal l, so that $\overline{\top} = \bot$, $\overline{\bot} = \top$, $\overline{x} = \neg x$ 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 $var(\neg x) = x$. A clause (resp. a term) is a finite disjunction (resp. conjunction) of literals. An XOR-clause is a finite XOR-disjunction of literals (the XOR connective is denoted by $\oplus$.)

Each element α of `QDAG` is called a `QDAG` formula. $Var(\alpha)$ denotes the set of free variables x 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 $\exists x$. The size $|\alpha|$ of a `QDAG` formula $|\alpha|$ is the number of nodes plus the number of arcs in the DAG.

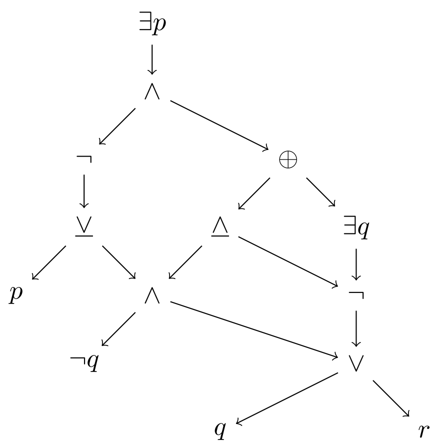

Figure 1: A `QDAG` formula.

Figure 1 presents a `QDAG` formula. Its set of free variables is $\{q, r\}$. The DAG rooted at the $\underline{\wedge}$ node is a `DAG` formula and the DAG rooted at the $\underline{\vee}$ node is a `DAG-NNF` formula.

In the following, we consider the next queries and transformations:

Definition 2 (queries) *Let $\mathcal{L} \subseteq$* `QDAG`.

- *$\mathcal{L}$ satisfies* **CO** *(resp.* **VA***) iff there exists a polytime algorithm that maps every formula α from $\mathcal{L}$ to 1 if α is consistent (resp. valid), and to 0 otherwise.*

- *$\mathcal{L}$ satisfies* **CE** *iff there exists a polytime algorithm that maps every formula α from $\mathcal{L}$ and every clause γ to 1 if $\alpha \models \gamma$ holds, and to 0 otherwise.*
- *$\mathcal{L}$ satisfies* **EQ** *(resp.* **SE***) iff there exists a polytime algorithm that maps every pair of formulae α, β from $\mathcal{L}$ to 1 if $\alpha \equiv \beta$ (resp. $\alpha \models \beta$) holds, and to 0 otherwise.*
- *$\mathcal{L}$ satisfies* **IM** *iff there exists a polytime algorithm that maps every formula α from $\mathcal{L}$ and every term γ to 1 if $\gamma \models \alpha$ holds, and to 0 otherwise.*
- *$\mathcal{L}$ satisfies* **CT** *iff there exists a polytime algorithm that maps every formula 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** *iff there exists a polynomial $p(.,.)$ and an algorithm that outputs all models of an arbitrary formula 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** *(model checking) iff there exists a polytime algorithm that maps every formula α from $\mathcal{L}$ and every interpretation ω over $Var(\alpha)$ (represented as a term) to 1 if ω is a model of α, and to 0 otherwise.*

Definition 3 (transformations) *Let $\mathcal{L} \subseteq$ `QDAG`.*

- *$\mathcal{L}$ satisfies* **CD** *iff there exists a polytime algorithm that maps every formula α from $\mathcal{L}$ and every consistent term γ to a formula from $\mathcal{L}$ that is logically equivalent to the conditioning $\alpha \mid \gamma$ of α on γ, i.e., the formula obtained by replacing each free occurrence of variable x of α by $\top$ (resp. $\bot$) if x (resp. $\neg x$) is a positive (resp. negative) literal of γ.*
- *$\mathcal{L}$ satisfies* **FO** *iff there exists a polytime algorithm that maps every formula α from $\mathcal{L}$ and every subset $\mathbf{X}$ of variables from PS to a formula 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**.
- *$\mathcal{L}$ satisfies $\wedge$***C** *(resp. $\vee$***C***) iff there exists a polytime algorithm that maps every finite set of formulae $\alpha_1, \ldots, \alpha_n$ from $\mathcal{L}$ to a formula of $\mathcal{L}$ that is logically equivalent to $\alpha_1 \wedge \ldots \wedge \alpha_n$ (resp. $\alpha_1 \vee \ldots \vee \alpha_n$.)*
- *$\mathcal{L}$ satisfies $\wedge$***BC** *(resp. $\vee$***BC***) iff there exists a polytime algorithm that maps every pair of formulae α and β from $\mathcal{L}$ to a formula of $\mathcal{L}$ that is logically equivalent to $\alpha \wedge \beta$ (resp. $\alpha \vee \beta$.)*
- *$\mathcal{L}$ satisfies $\neg$***C** *iff there exists a polytime algorithm that maps every formula α from $\mathcal{L}$ to a formula of $\mathcal{L}$ logically equivalent to $\neg\alpha$.*

We also consider the following notions of expressiveness and succinctness:

Definition 4 (expressiveness) *Let $\mathcal{L}_1$ and $\mathcal{L}_2$ be two subsets of* `QDAG`*. $\mathcal{L}_1$ is at least as expressive as $\mathcal{L}_2$, denoted $\mathcal{L}_1 \leq_e \mathcal{L}_2$, iff for every formula $\alpha \in \mathcal{L}_2$, there exists an equivalent formula $\beta \in \mathcal{L}_1$.*

Definition 5 (succinctness) *Let $\mathcal{L}_1$ and $\mathcal{L}_2$ be two subsets of* `QDAG`*. $\mathcal{L}_1$ is at least as succinct as $\mathcal{L}_2$, denoted $\mathcal{L}_1 \leq_s \mathcal{L}_2$, iff there exists a polynomial p such that for every formula $\alpha \in \mathcal{L}_2$, there exists an equivalent formula $\beta \in \mathcal{L}_1$ where $|\beta| \leq p(|\alpha|)$.*

Finally, we take advantage of the following restriction of the succinctness relation:

Definition 6 (polynomial translation) *Let $\mathcal{L}_1$ and $\mathcal{L}_2$ be two subsets of* `QDAG`*. $\mathcal{L}_1$ is said to be* polynomially translatable *into $\mathcal{L}_2$, noted $\mathcal{L}_1 \geq_p \mathcal{L}_2$, iff there exists a polytime algorithm f such that for every $\alpha \in \mathcal{L}_1$, we have $f(\alpha) \in \mathcal{L}_2$ and $f(\alpha) \equiv \alpha$.*

Whenever $\mathcal{L}_1$ is polynomially translatable into $\mathcal{L}_2$, every query which is supported in polynomial time in $\mathcal{L}_2$ also is supported in polynomial time in $\mathcal{L}_1$; and conversely, every query which is not supported in polynomial time in $\mathcal{L}_1$ unless P = NP (resp. unless the polynomial hierarchy PH collapses) cannot be supported in polynomial time in $\mathcal{L}_2$, unless P = NP (resp. unless PH collapses.)

$\sim_e$ is the symmetric part of $\leq_e$ defined by $\mathcal{L}_1 \sim_e \mathcal{L}_2$ iff $\mathcal{L}_1 \leq_e \mathcal{L}_2$ and $\mathcal{L}_2 \leq_e \mathcal{L}_1$. $<_e$ is the asymmetric part of $\leq_e$ defined by $\mathcal{L}_1 <_e \mathcal{L}_2$ iff $\mathcal{L}_1 \leq_e \mathcal{L}_2$ and $\mathcal{L}_2 \not\leq_e \mathcal{L}_1$. Similarly, $\sim_s$ (resp. $\sim_p$) is the symmetric part of $\leq_s$ (resp. $\leq_p$.) $<_s$ (resp. $<_p$) is the asymmetric part of $\leq_s$ (resp. $\leq_p$.) When $\mathcal{L}_1 \sim_p \mathcal{L}_2$, $\mathcal{L}_1$ and $\mathcal{L}_2$ are said to be polynomially equivalent. Obviously enough, polynomially equivalent fragments are equally efficient (and succinct) and possess the same set of tractable queries and transformations.

We are now ready to present disjunctive closures. Intuitively, a closure principle applied to a propositional fragment $\mathcal{L}$ defines a new propositional language, called a closure of $\mathcal{L}$, through the application of "operators" (i.e., connectives or quantifications.) The resulting closure is said to be *disjunctive* when the operators are among $\vee$ and $\exists x$ with $x \in PS$. Formally:

Definition 7 (disjunctive closures) *Let $\mathcal{L} \subseteq$ `QDAG` and $\triangle \subseteq \{\vee, \exists\}$. The* closure *$\mathcal{L}[\triangle]$ of $\mathcal{L}$ by $\triangle$ is the subset of* `QDAG` *inductively defined as follows:*[1]

1. *if $\alpha \in \mathcal{L}$, then $\alpha \in \mathcal{L}[\triangle]$,*
2. *if $\vee \in \Delta$ and $\alpha_i \in \mathcal{L}[\triangle]$ for each $i \in 1, \ldots, n$, then $\vee(\alpha_1, \ldots, \alpha_n) \in \mathcal{L}[\triangle]$,*
3. *if $\exists \in \Delta$, $x \in PS$, and $\alpha \in \mathcal{L}[\triangle]$, then $\exists x.\alpha \in \mathcal{L}[\triangle]$.*

Thus, an element of $\mathcal{L}[\triangle]$ can be viewed as a "tree" which "internal nodes" are labelled by quantifications of the form $\exists x$ or by $\vee$ and its "leaf nodes" are labelled by elements of $\mathcal{L}$. Accordingly, the formulae α_i considered in item 2. of Definition 7 do not share any common subgraphs.

The set $\mathcal{D}(\mathcal{L})$ (resp. $\bigvee(\mathcal{L})$, $\exists(\mathcal{L})$) of all disjunctive (resp. disjunction, existential) closures of a subset $\mathcal{L}$ of `QDAG` is defined inductively by:

- $\mathcal{L}$ belongs to $\mathcal{D}(\mathcal{L})$ (resp. $\bigvee(\mathcal{L})$, $\exists(\mathcal{L})$);
- If $\mathcal{L}$ belongs to $\mathcal{D}(\mathcal{L})$ (resp. $\bigvee(\mathcal{L})$, $\exists(\mathcal{L})$) and $\triangle \subseteq \{\vee, \exists\}$ (resp. $\triangle \subseteq \{\vee\}$, $\triangle \subseteq \{\exists\}$), then $\mathcal{L}[\triangle]$ belongs to $\mathcal{D}(\mathcal{L})$ (resp. $\bigvee(\mathcal{L})$, $\exists(\mathcal{L})$.)

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

[Fargier and Marquis, 2008a] provide several general-scope characterization results for disjunctive closures. Especially, they show that for a given $\mathcal{L} \subseteq$ DAG, only three disjunctive closures are worth to be considered since $\mathcal{L}[\exists][\exists] = \mathcal{L}[\exists]$, $\mathcal{L}[\vee][\vee] = \mathcal{L}[\vee]$ and $\mathcal{L}[\exists][\vee] \sim_p \mathcal{L}[\vee][\exists] \sim_p \mathcal{L}[\vee, \exists]$ (see item 4. of Proposition 1 and item 1. of Proposition 2 in [Fargier and Marquis, 2008a].) They also study the disjunctive closures of the languages $\mathtt{OBDD}_<$, DNF, DNNF, CNF, PI, IP, MODS considered in [Darwiche and Marquis, 2002].

3 Analyzing New Existential Closures

[Fargier and Marquis, 2008b] focus on the disjunction closures of the following incomplete fragments:

Definition 8 (some incomplete fragments)

- KROM-C *is the subset of* CNF *consisting of formulae in which each clause is binary, i.e., it contains at most two literals.*
- HORN-C *is the subset of* CNF *consisting of formulae in which each clause is Horn, i.e., it contains at most one positive literal.*
- K/H-C *is the union of* KROM-C *and* HORN-C.
- renH-C *is the subset of* CNF *consisting of formulae α for which there exists a subset V of $Var(\alpha)$ (called a Horn renaming for α) such that the formula $V(\alpha)$ obtained by substituting in α every literal l over V by its complementary literal $\bar{l}$ is a* HORN-C *formula.*[2]
- AFF *is the subset of* DAG *consisting of conjunctions of XOR clauses.*

In the following, we complete the results provided in [Fargier and Marquis, 2008b] by studying the queries, transformations, expressiveness and succinctness of the existential closures of the propositional fragments considered in [Fargier and Marquis, 2008b], i.e., we consider the languages KROM-C[$\exists$], HORN-C[$\exists$], K/H-C[$\exists$], renH-C[$\exists$], AFF[$\exists$], KROM-C[$\vee, \exists$], HORN-C[$\vee, \exists$], K/H-C[$\vee, \exists$], renH-C[$\vee, \exists$], AFF[$\vee, \exists$].

Figure 2 presents a HORN $-$ C[$\vee, \exists$] formula equivalent to the CNF formula $(\neg p \vee \neg r) \wedge (\neg r \vee \neg s) \wedge (\neg q \vee \neg s)$.

We know that each of KROM-C and AFF satisfies **FO** [Fargier and Marquis, 2008b]. Furthermore, for any $\mathcal{L} \subseteq$ DAG, if $\mathcal{L}$ satisfies **FO**, then $\mathcal{L}[\vee]$ satisfies **FO** as well. As a consequence, we immediately get that

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

Since KROM-C, AFF, KROM-C[$\vee$], and AFF[$\vee$] have been studied in [Fargier and Marquis, 2008b], we mainly focus on HORN-C[$\exists$], K/H-C[$\exists$], renH-C[$\exists$], HORN-C[$\vee, \exists$], K/H-C[$\vee, \exists$], renH-C[$\vee, \exists$] in the following.

[2]Note that there exists linear time algorithms for recognizing renH-C formulae (see e.g. [del Val, 2000]); furthermore, such recognition algorithms typically give a Horn renaming when it exists.

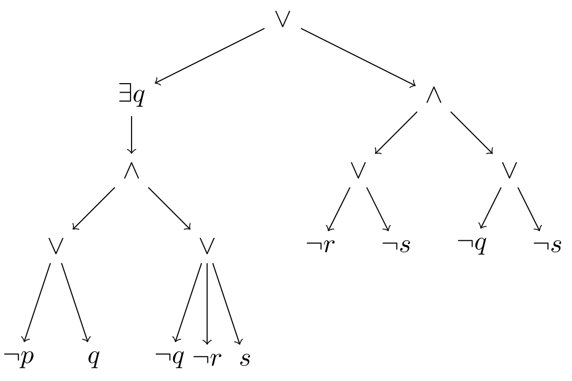

Figure 2: A HORN $-$ C[$\vee, \exists$] formula.

$\mathcal{L}$	CO	VA	CE	IM	EQ	SE	CT	ME	MC
renH-C[$\vee, \exists$]	√	○	√	○	○	○	○	√	√
K/H-C[$\vee, \exists$]	√	○	√	○	○	○	○	√	√
HORN-C[$\vee, \exists$]	√	○	√	○	○	○	○	√	√
renH-C[$\exists$]	√	√	√	√	○	○	○	√	√
K/H-C[$\exists$]	√	√	√	√	○	○	○	√	√
HORN-C[$\exists$]	√	√	√	√	○	○	○	√	√

Table 1: Queries. √ means "satisfies" and ○ means "does not satisfy unless P = NP."

As to queries and transformations, we have obtained the following results:

Proposition 1 *The results given in Table 1 and in Table 2 hold.*

As to expressiveness, it turns out that for any language $\mathcal{L}$ among K/H-C, HORN-C, renH-C, K/H-C[$\vee$], HORN-C[$\vee$], renH-C[$\vee$], going from $\mathcal{L}$ to $\mathcal{L}[\exists]$ does not lead to a shift:

Proposition 2 *For every $\mathcal{L}$ among* K/H-C, HORN-C, renH-C, K/H-C[$\vee$], HORN-C[$\vee$], renH-C[$\vee$], *we have* $\mathcal{L} \sim_e \mathcal{L}[\exists]$.

As a consequence (see [Fargier and Marquis, 2008b]), we have HORN-C[$\vee, \exists$] $\sim_e$ K/H-C[$\vee, \exists$] $\sim_e$ renH-C[$\vee, \exists$]: these three languages are just equally expressive since they are complete for propositional logic (just observe that the complete language DNF is a subset of each of them.) Contrastingly, each of HORN-C[$\exists$], K/H-C[$\exists$], renH-C[$\exists$] is strictly less expressive than any complete language.

$\mathcal{L}$	CD	FO	SFO	$\wedge$C	$\wedge$BC	$\vee$C	$\vee$BC	$\neg$C
renH-C[$\vee, \exists$]	√	√	√	○	○	√	√	○
K/H-C[$\vee, \exists$]	√	√	√	○	○	√	√	○
HORN-C[$\vee, \exists$]	√	√	√	○	√	√	√	○
renH-C[$\exists$]	√	√	√	!	!	!	!	!
K/H-C[$\exists$]	√	√	√	○	○	!	!	!
HORN-C[$\exists$]	√	√	√	√	√	!	!	!

Table 2: Transformations. √ means "satisfies," while ○ means "does not satisfy unless P=NP." ! means that the transformation is not always feasible within the fragment.

Furthermore, we have:

$$\texttt{renH-C}[\exists] <_e \texttt{K/H-C}[\exists] <_e \texttt{HORN-C}[\exists]$$
$$\texttt{K/H-C}[\exists] <_e \texttt{KROM-C}[\exists]$$

while HORN−C[∃] and KROM−C[∃] are incomparable w.r.t. $\leq_e$ (and AFF[∃], which is polynomially equivalent to AFF, is incomparable w.r.t. $\leq_e$ with any of the three incomplete fragments above.)

As to succinctness, the picture is rather different. Typically going from $\mathcal{L}$ to $\mathcal{L}[\exists]$ leads to a a succinctness increase:

Proposition 3

1. *None of* CNF *or* DNF *is at least at succinct as any of* HORN−C[∃], K/H−C[∃] *or* renH−C[∃].
2. AC^3*, the language containing all disjunctions of* CNF *formulae and all conjunctions of* DNF *formulae, is not at least as succinct as any of* HORN−C[∨, ∃], K/H−C[∨, ∃] *and* renH−C[∨, ∃].

Since HORN−C (resp. K/H−C, renH−C) is a subset of both CNF and HORN−C[∃] (resp. K/H−C[∃], renH−C[∃]), item 1. of Proposition 3 shows that going from $\mathcal{L}$ to $\mathcal{L}[\exists]$ for any of these three languages leads to a strictly more succinct language. Furthermore, since HORN−C[∨] (resp. K/H−C[∨], renH−C[∨]) is a subset of both AC^3 and HORN−C[∨, ∃] (resp. K/H−C[∨, ∃], renH−C[∨, ∃]), item 2. of Proposition 3 shows that a similar conclusion can be drawn for those three languages, namely HORN−C[∨] (resp. K/H−C[∨], renH−C[∨]) is strictly less succinct than HORN−C[∨, ∃] (resp. K/H−C[∨, ∃], renH−C[∨, ∃].) As a by-product, this shows that for sure, none of HORN−C[∨], K/H−C[∨], renH−C[∨] satisfies **FO**, an issue left open in [Fargier and Marquis, 2008b].

Finally, we have also compared the disjunction closures of KROM−C, HORN−C, K/H−C, renH−C, AFF, PI with the two main subsets of DNNF, namely the set d−DNNF of deterministic DNNFs and the set SDNNF of structured DNNFs (the union for all vtrees T of all DNNF languages DNNF$_\mathrm{T}$ respecting T) (see [Pipatsrisawat and Darwiche, 2008] for details.) What makes those subsets so significant is that every available compilation algorithm which outputs a DNNF formula actually computes a d−DNNF formula [Darwiche, 2001; 2004] or a SDNNF formula [Pipatsrisawat and Darwiche, 2008; 2010]. First of all, as an easy consequence of the fact that DNNF is not at least as succinct as DNF, unless PH collapses, we have that:

Proposition 4 d−DNNF *is not at least as succinct as any language among the existential closures of* KROM−C[∨], HORN−C[∨], K/H−C[∨], renH−C[∨], AFF[∨], PI[∨], *and* OBDD$_<$[∨], *unless* PH *collapses.*

This shows those existential closures as interesting alternatives to d−DNNF for applications where **FO**, ∨**C** (and for some of them ∧**BC**) are required since such closures offer them while d−DNNF does not (the price to be paid is that **VA**, **IM** and **CT** are lost while they are offered by d−DNNF.)

As to SDNNF, we have got that:

Proposition 5 SDNNF *is not at least as succinct as any language among the disjunctive closures of* KROM−C, HORN−C, K/H−C, renH−C, AFF, PI, *and* CNF.

Thus, every language that is at most as succinct as SDNNF is not at least as succinct as any disjunctive closure of KROM−C, HORN−C, K/H−C, renH−C, AFF, PI, CNF. This includes in particular OBDD$_<$, OBDD$_<$[∨], the language O−DDG of Ordered Decomposable Decision Diagrams [Fargier and Marquis, 2006], and the language AOMDD of AND/OR Multi-Valued Decision Diagrams (restricted to propositional variables) [Mateescu *et al.*, 2008] (or equivalently the language SO−DDG of Strongly Ordered Decomposable Decision Diagrams [Fargier and Marquis, 2006].)

4 Conclusion and Perspectives

In this paper, we have studied the existential closures of both incomplete languages (KROM−C, HORN−C, K/H−C, renH−C, and AFF) and complete languages (the corresponding disjunction closures KROM−C[∨], HORN−C[∨], K/H−C[∨], renH−C[∨], and AFF[∨].) The results given above show that for each complete language $\mathcal{L}$ under consideration, the corresponding existential closure $\mathcal{L}[\exists]$ is at least as good as $\mathcal{L}$ as a target language for KC. Indeed, $\mathcal{L}$ and $\mathcal{L}[\exists]$ satisfy the same queries and when $\mathcal{L}$ does not satisfy **FO**, $\mathcal{L}[\exists]$ offers it for free and is equally expressive but strictly more succinct than $\mathcal{L}$. For the incomplete languages considered here, namely KROM−C, HORN−C, K/H−C, renH−C, AFF, the same conclusions can be drawn as to transformations, expressiveness and succinctness; the price to be paid for getting **FO** for free and obtaining a strictly more succinct language is paid by the loss of the **EQ** and **SE** queries when the existential closures of HORN−C, K/H−C, renH−C are considered.

Our study also shows $\mathtt{HORN}-\mathtt{C}[\vee,\exists]$ as a valuable complete target language for the KC purpose. Indeed, $\mathtt{HORN}-\mathtt{C}[\vee,\exists]$ satisfies the same queries and transformations as DNNF$_\mathrm{T}$ or AFF[∨] (which is polynomially equivalent to AFF[∨, ∃].) Especially, $\mathtt{HORN}-\mathtt{C}[\vee,\exists]$ satisfies ∧**BC** and this paves the way for bottom-up compilation algorithms targeting $\mathtt{HORN}-\mathtt{C}[\vee,\exists]$. As noted in [Pipatsrisawat and Darwiche, 2008], this is important for applications from formal verification based on unbounded model checking which require bottom-up, incremental compilation of formulae, where pieces of the knowledge base are compiled independently and then conjoined together.[3] Furthermore, $\mathtt{HORN}-\mathtt{C}[\vee,\exists]$ guarantees some polynomial-sized representations for families of propositional formulae, like the CNF representations of circular bit shift functions (resp. the CNF formulae $\alpha_n = \bigwedge_{i=1}^{n}(\neg x_i \vee \neg y_i)$), which have only exponential-sized SDNNF representations (resp. exponential-sized AFF[∨] representations, see [Fargier and Marquis, 2008b].) Compared to d−DNNF, while it does not satisfy the queries **VA**, **IM** and **CT**, $\mathtt{HORN}-\mathtt{C}[\vee,\exists]$ offers the transformations **FO**, ∨**C** and ∧**BC**. Finally, d−DNNF is not at least as succinct as $\mathtt{HORN}-\mathtt{C}[\vee,\exists]$ unless PH collapses.

From the practical side, it is important to note that some empirical evidence in favour of $\mathtt{HORN}-\mathtt{C}[\vee,\exists]$ compila-

[3]Since $\mathtt{HORN}-\mathtt{C}[\vee,\exists]$ satisfies **CO** and **CL** (i.e., every propositional clause has a polynomial-sized representation in the language), getting ∧**BC** is optimal in the sense that no propositional language can satisfy both ∧**C**, **CO**, and **CL**, unless P = NP.

tions and $\mathtt{renH} - \mathtt{C}[\vee, \exists]$ already exists. On the one hand, [Nishimura *et al.*, 2004] (resp. [Samer and Szeider, 2008]) have shown that the problem of determining whether a given CNF formula α has a strong HORN−C-backdoor set (resp. a HORN−C-backdoor tree) containing at most k variables (resp. leaves) is fixed-parameter tractable with parameter k. Interestingly, the algorithm given in [Samer and Szeider, 2008] can be used to determine "efficiently"(i.e., for sufficiently "small" k) whether a $\mathtt{HORN} - \mathtt{C}[\vee]$ compilation of reasonable size (i.e., linear in k and the size of α) exists. As mentioned in [Samer and Szeider, 2008]: "There is some empirical evidence that real-world instances actually have small backdoor sets". On the other hand, [Boufkhad *et al.*, 1997] present some compilation algorithms targeting respectively $\mathtt{HORN} - \mathtt{C}[\vee]$ and $\mathtt{renH} - \mathtt{C}[\vee]$, and evaluate them on a number of benchmarks. While such results show the feasibility of $\mathtt{HORN} - \mathtt{C}[\vee, \exists]$ compilation and $\mathtt{renH} - \mathtt{C}[\vee, \exists]$ compilation, we can hardly use those results to compare the two target fragments $\mathtt{HORN} - \mathtt{C}[\vee, \exists]$ and $\mathtt{renH} - \mathtt{C}[\vee, \exists]$ with $\mathtt{OBDD}_<$ and DNNF for which some experimental results are also available. Indeed, the compilation algorithms given in [Boufkhad *et al.*, 1997] 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. Accordingly, the main perspective of this work is to develop and evaluate compilation algorithms based on a CDCL SAT solver, and which target $\mathtt{HORN} - \mathtt{C}[\vee, \exists]$ and $\mathtt{renH} - \mathtt{C}[\vee, \exists]$.

Acknowledgments

The author would like to thank the anonymous reviewers for their helpful comments.

References

[Boufkhad *et al.*, 1997] Y. Boufkhad, E. Grégoire, P. Marquis, B. Mazure, and L. Saïs. Tractable cover compilations. In *Proc. of IJCAI'97*, pages 122–127, 1997.

[Cadoli and Donini, 1998] M. Cadoli and F.M. Donini. A survey on knowledge compilation. *AI Communications*, 10(3–4):137–150, 1998.

[Darwiche and Marquis, 2002] A. Darwiche and P. Marquis. A knowledge compilation map. *Journal of Artificial Intelligence Research*, 17:229–264, 2002.

[Darwiche, 2001] A. Darwiche. Decomposable negation normal form. *Journal of the ACM*, 48(4):608–647, 2001.

[Darwiche, 2004] A. Darwiche. New advances in compiling cnf into decomposable negation normal form. In *Proc. of ECAI'04*, pages 328–332, 2004.

[del Val, 1994] A. del Val. Tractable databases: How to make propositional unit resolution complete through compilation. In *Proc. of KR'94*, pages 551–561, 1994.

[del Val, 2000] A. del Val. On 2-SAT and Renamable Horn. In *Proc. of AAAI'00*, pages 279–284, 2000.

[Fargier and Marquis, 2006] H. Fargier and P. Marquis. On the use of partially ordered decision graphs in knowledge compilation and quantified Boolean formulae. In *Proc. of AAAI'06*, pages 42–47, 2006.

[Fargier and Marquis, 2008a] H. Fargier and P. Marquis. Extending the knowledge compilation map: Closure principles. In *Proc. of ECAI'08*, pages 50–54, 2008.

[Fargier and Marquis, 2008b] H. Fargier and P. Marquis. Extending the knowledge compilation map: Krom, Horn, affine and beyond. In *Proc. of AAAI'08*, pages 442–447, 2008.

[Fargier and Marquis, 2009] H. Fargier and P. Marquis. Knowledge compilation properties of trees-of-bdds, revisited. In *Proc. of IJCAI'09*, pages 772–777, 2009.

[Gogic *et al.*, 1995] G. Gogic, H.A. Kautz, Ch.H. Papadimitriou, and B. Selman. The comparative linguistics of knowledge representation. In *Proc. of IJCAI'95*, pages 862–869, 1995.

[Horn, 1951] A. Horn. On sentences which are true of direct unions of algebras. *Journal of Symbolic Logic*, 16:14–21, 1951.

[Krom, 1970] M.R. Krom. The decision problem for formulas in prenex conjunctive normal form with binary disjunctions. *Journal of Symbolic Logic*, 35:210–216, 1970.

[Mateescu *et al.*, 2008] R. Mateescu, R. Dechter, and R. Marinescu. And/or multi-valued decision diagrams (aomdds) for graphical models. *Journal of Artificial Intelligence Research*, 33:465–519, 2008.

[Nishimura *et al.*, 2004] N. Nishimura, P. Ragde, and S. Szeider. Detecting backdoor sets with respect to horn and binary clauses. In *Proc. of SAT'04*, 2004.

[Pipatsrisawat and Darwiche, 2008] K. Pipatsrisawat and A. Darwiche. New compilation languages based on structured decomposability. In *Proc. of AAAI'08*, pages 517–522, 2008.

[Pipatsrisawat and Darwiche, 2010] K. Pipatsrisawat and A. Darwiche. Top-down algorithms for constructing structured dnnf: Theoretical and practical implications. In *Proc. of ECAI'10*, pages 3–8, 2010.

[Samer and Szeider, 2008] M. Samer and S. Szeider. Backdoor trees. In *Proc. of AAAI'08*, pages 363–368, 2008.

[Schaefer, 1978] Th. J. Schaefer. The complexity of satisfiability problems. In *Proc. of STOC'78*, pages 216–226, 1978.

[Schrag, 1996] R. Schrag. Compilation for critically constrained knowledge bases. In *Proc. of AAAI'96*, pages 510–515, 1996.

[Selman and Kautz, 1996] B. Selman and H.A. Kautz. Knowledge compilation and theory approximation. *Journal of the ACM*, 43:193–224, 1996.

[Subbarayan *et al.*, 2007] S. Subbarayan, L. Bordeaux, and Y. Hamadi. Knowledge compilation properties of tree-of-BDDs. In *Proc. of AAAI'07*, pages 502–507, 2007.

[Wachter and Haenni, 2006] M. Wachter and R. Haenni. Propositional DAGs: A new graph-based language for representing Boolean functions. In *Proc. of KR'06*, pages 277–285, 2006.