# Extending the Knowledge Compilation Map: Closure Principles

**Hélène Fargier**[1] and **Pierre Marquis**[2]

**Abstract.**

We extend the knowledge compilation map introduced by Darwiche and Marquis with new propositional fragments obtained by applying closure principles to several fragments studied so far. We investigate two closure principles: disjunction and implicit forgetting (i.e., existential quantification). Each introduced fragment is evaluated w.r.t. several criteria, including the complexity of basic queries and transformations, and its spatial efficiency is also analyzed.

## 1 INTRODUCTION

This paper is concerned with knowledge compilation (KC). The key idea underlying KC is to pre-process parts of the available data (i.e., turning them into a compiled form) for improving the efficiency of some computational tasks (see among others [2, 1, 10, 4]). A research line in KC [7, 3] addresses the following important issue: How to choose a target language for knowledge compilation? In [3], the authors argue that the choice of a target language for a compilation purpose in the propositional case must be based both on the set of queries and transformations which can be achieved in polynomial time when the data to be exploited are represented in the language, as well as the spatial efficiency of the language (i.e., its ability to represent data using little space). Thus, the KC map reported in [3] is an evaluation of dozen of significant propositional languages (called propositional fragments) w.r.t. several dimensions: the spatial efficiency (i.e., succinctness) of the fragment and the class of queries and transformations it supports in polynomial time.

The basic queries considered in [3] include tests for consistency, validity, implicates (clausal entailment), implicants, equivalence, sentential entailment, counting and enumerating theory models (**CO**, **VA**, **CE**, **EQ**, **SE**, **IM**, **CT**, **ME**). The basic transformations are conditioning (**CD**), (possibly bounded) closures under the connectives $\wedge$, $\vee$, and $\neg$ ( $\wedge\mathbf{C}$, $\wedge\mathbf{BC}$, $\vee\mathbf{C}$, $\vee\mathbf{BC}$, $\neg\mathbf{C}$) and (possibly bounded) forgetting which can be viewed as a closure operation under existential quantification (**FO**, **SFO**).

The KC map reported in [3] has already been extended to new propositional languages, queries and transformations in [12, 5, 11]. In this paper, we extend the KC map with new propositional fragments obtained by applying closure principles to several fragments studied so far. Intuitively, a closure principle is a way to define a new propositional fragment from a previous one. In this paper, we investigate in detail two disjunctive closure principles, disjunction ($\vee$) and implicit forgetting ($\exists$), and their combinations. Roughly speaking, the disjunction principle when applied to a fragment $\mathcal{C}$ leads to a fragment $\mathcal{C}[\vee]$ which allows disjunctions of formulas from $\mathcal{C}$, while implicit forgetting applied to a fragment $\mathcal{C}$ leads to a fragment $\mathcal{C}[\exists]$ which allows existentially quantified formulas from $\mathcal{C}$. Obviously enough, whatever $\mathcal{C}$, $\mathcal{C}[\vee]$ satisfies polytime closure under $\vee$ ($\vee$**C**) and $\mathcal{C}[\exists]$ satisfies polytime forgetting (**FO**). Applying any/both of those two principles may lead to new fragments, which can prove strictly more succinct than the underlying fragment $\mathcal{C}$; interestingly, this gain in efficiency does not lead to a complexity shift w.r.t. the main queries and transformations; indeed, among other things, our results show that whenever $\mathcal{C}$ satisfies **CO** (resp. **CD**), then $\mathcal{C}[\vee]$ and $\mathcal{C}[\exists]$ satisfy **CO** (resp. **CD**).

The remainder of this paper is organized as follows. In Section 2, we define the language of quantified propositional DAGs. In Section 3, we extend the usual notions of queries, transformations and succinctness to this language. In Section 4, we introduce the general principle of closure by a connective or a quantification before focusing on the disjunctive closures of the fragments considered in [3] and studying their attractivity for KC, thus extending the KC map. In Section 5, we discuss the results. Finally, Section 6 concludes the paper.

## 2 A GLIMPSE AT QUANTIFIED PDAGS

All the propositional fragments we consider in this paper are subsets of the following language of quantified propositional DAGs $Q\mathsf{PDAG}$:

**Definition 1 (quantified PDAGs)** *Let $PS$ be a denumerable set of propositional variables (also called atoms).*

- $Q\mathsf{PDAG}$ *is the set of all finite, single-rooted DAGs $\alpha$ (called* formulas*) where each leaf node is labeled by a literal over $PS$ or one of the two Boolean constants $\top$ or $\bot$, and each internal node is labeled by $\wedge$ or $\vee$ and has arbitrarily many children or is labeled by $\neg$, $\exists x$ or $\forall x$ (where $x \in PS$) and has just one child.*
- $Q_p\mathsf{PDAG}$ *is the subset of all* proper *formulas of $Q\mathsf{PDAG}$, where a formula $\alpha$ is proper iff for every literal $l = x$ or $l = \neg x$ labelling a leaf of $\alpha$, at most one path from the root of $\alpha$ to this leaf contains quantifications of the form $\exists x$ or $\forall x$, and if such a path exists, it is the unique path from the root of $\alpha$ to the leaf.*

Restricting the language $Q\mathsf{PDAG}$ to proper formulas $\alpha$ ensures that every occurrence of a variable $x$ corresponding to a literal at a leaf of $\alpha$ depends on at most one quantification on $x$, and is either free or bound. As a consequence (among others), conditioning a proper formula can be achieved as usual (without requiring any duplication of nodes).

[1] IRIT-CNRS, Université Paul Sabatier, France, email: fargier@irit.fr
[2] Université Lille-Nord de France, Artois, CRIL UMR CNRS 8188, France, email: marquis@cril.univ-artois.fr

PDAG [12] is the subset of $Q_p$PDAG obtained by removing the possibility to have internal nodes labeled by $\exists$ or $\forall$; PDAG-NNF [3] (resp. $\exists$PDAG-NNF, resp. $\forall$PDAG-NNF) is the subset of $Q_p$PDAG obtained by removing the possibility to have internal nodes labeled by $\neg$, $\exists$ or $\forall$ (resp. $\neg$, $\forall$, resp. $\neg$, $\exists$). Distinguished formulas from $Q$PDAG are the literals over $PS$; if $V$ is any subset of $PS$, $L_V$ denotes the set of all literals built over $V$, i.e., $\{x, \neg x \mid x \in V\}$. If a literal $l$ of $L_{PS}$ is an atom $x$ from $PS$, it is said to be a positive literal; otherwise it has the form $\neg x$ with $x \in PS$ and it is said to be a negative literal. If $l$ is a literal built up from the atom $x$, we have $var(l) = x$. A clause (resp. a term) is a (finite) disjunction (resp. conjunction) of literals or the constant $\bot$ (resp. $\top$). The size $|\alpha|$ of any $Q$PDAG formula $\alpha$ is the number of nodes plus the number of arcs in $\alpha$. The set $Var(\alpha)$ of free variables of a $Q_p$PDAG formula $\alpha$ is defined in the standard way.

Let $I$ be an interpretation over $PS$ (i.e., a total function from $PS$ to $BOOL = \{0, 1\}$). The semantics of a $Q$PDAG formula $\alpha$ in $I$ is the truth value from $BOOL$ defined inductively in the standard way; the notions of model, logical consequence ($\models$) and logical equivalence ($\equiv$) are also as usual.

Finally, if $\alpha \in Q$PDAG and $X = \{x_1, \ldots, x_n\} \subseteq PS$, then $\exists X.\alpha$ (resp. $\forall X.\alpha$) is a short for $\exists x_1.(\exists x_2.(...\exists x_n.\alpha)...)$ (resp. $\forall x_1.(\forall x_2.(...\forall x_n.\alpha)...)$) (this notation is well-founded since whatever the chosen ordering on $X$, the resulting formulas are logically equivalent).

## 3 QUERIES, TRANSFORMATIONS, AND SUCCINCTNESS

The following queries **CO**, **VA**, **CE**, **EQ**, **SE**, **IM**, **CT**, **ME** for PDAG-NNF formulas have been considered in [3]; their importance is discussed in depth in [3], so we refrain from recalling it here; we extend them to $Q_p$PDAG formulas and add to them the **MC** query (model checking), which is trivial for PDAG formulas (every formula from PDAG satisfies **MC**), but not for $Q_p$PDAG formulas.

**Definition 2 (queries)** *Let $\mathcal{C}$ denote any subset of $Q_p$PDAG.*

- *$\mathcal{C}$ satisfies* **CO** *(resp.* **VA***) iff there exists a polytime algorithm that maps every formula $\alpha$ from $\mathcal{C}$ to 1 if $\alpha$ is consistent (resp. valid), and to 0 otherwise.*
- *$\mathcal{C}$ satisfies* **MC** *iff there exists a polytime algorithm that maps every formula $\alpha$ from $\mathcal{C}$ and every interpretation $I$ over $Var(\alpha)$ to 1 if $I$ is a model of $\alpha$, and to 0 otherwise.*
- *$\mathcal{C}$ satisfies* **CE** *iff there exists a polytime algorithm that maps every formula $\alpha$ from $\mathcal{C}$ and every clause $\gamma$ to 1 if $\alpha \models \gamma$ holds, and to 0 otherwise.*
- *$\mathcal{C}$ satisfies* **EQ** *(resp.* **SE***) iff there exists a polytime algorithm that maps every pair of formulas $\alpha$, $\beta$ from $\mathcal{C}$ to 1 if $\alpha \equiv \beta$ (resp. $\alpha \models \beta$) holds, and to 0 otherwise.*
- *$\mathcal{C}$ satisfies* **IM** *iff there exists a polytime algorithm that maps every formula $\alpha$ from $\mathcal{C}$ and every term $\gamma$ to 1 if $\gamma \models \alpha$ holds, and to 0 otherwise.*
- *$\mathcal{C}$ satisfies* **CT** *iff there exists a polytime algorithm that maps every formula $\alpha$ from $\mathcal{C}$ to a nonnegative integer that represents the number of models of $\alpha$ over $Var(\alpha)$ (in binary notation).*
- *$\mathcal{C}$ satisfies* **ME** *iff there exists a polynomial $p(.,.)$ and an algorithm that outputs all models of an arbitrary formula $\alpha$ from $\mathcal{C}$ in time $p(n, m)$, where $n$ is the size of $\alpha$ and $m$ is the number of its models (over $Var(\alpha)$).*

The following transformations for PDAG-NNF formulas have been considered in [3]; again, we extend them to $Q_p$PDAG formulas:

**Definition 3 (transformations)** *Let $\mathcal{C}$ denote any subset of $Q_p$PDAG.*

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

Finally, the following notion of succinctness (modeled as a pre-order over propositional fragments) has been considered in [3]; we also extend it to $Q$PDAG formulas:

**Definition 4 (succinctness)** *Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be two subsets of $Q$PDAG. $\mathcal{C}_1$ is at least as succinct as $\mathcal{C}_2$, denoted $\mathcal{C}_1 \leq_s \mathcal{C}_2$, iff there exists a polynomial $p$ such that for every formula $\alpha \in \mathcal{C}_2$, there exists an equivalent formula $\beta \in \mathcal{C}_1$ where $|\beta| \leq p(|\alpha|)$.*

$\sim_s$ is the symmetric part of $\leq_s$ defined by $\mathcal{C}_1 \sim_s \mathcal{C}_2$ iff $\mathcal{C}_1 \leq_s \mathcal{C}_2$ and $\mathcal{C}_2 \leq_s \mathcal{C}_1$. $<_s$ is the asymmetric part of $\leq_s$ defined by $\mathcal{C}_1 <_s \mathcal{C}_2$ iff $\mathcal{C}_1 \leq_s \mathcal{C}_2$ and $\mathcal{C}_2 \not\leq_s \mathcal{C}_1$.

## 4 EXTENDING THE KC MAP BY DISJUNCTIVE CLOSURES

### 4.1 Closure Principles

Intuitively, a closure principle is a way to define a new propositional fragment starting from a previous one, through the application of "operators" (i.e., connectives or quantifications):[3]

**Definition 5 (closures)** *Let $\mathcal{C}$ be a subset of $Q$PDAG and $\triangle$ be any finite subset of $\{\vee, \wedge, \neg, \exists, \forall\}$. $\mathcal{C}[\triangle]$ is the subset of $Q$PDAG inductively defined as follows:*[4]

- *if $\alpha \in \mathcal{C}$, then $\alpha \in \mathcal{C}[\triangle]$,*
- *if $\delta \in \triangle \cap \{\vee, \wedge\}$, and $\alpha_i \in \mathcal{C}[\triangle]$ with $i \in 1 \ldots n$ and $n > 0$, then $\delta(\alpha_1, \ldots, \alpha_n) \in \mathcal{C}[\triangle]$,*
- *if $\neg \in \triangle$ and $\alpha \in \mathcal{C}[\triangle]$, then $\neg\alpha \in \mathcal{C}[\triangle]$,*
- *if $\delta \in \triangle \cap \{\forall, \exists\}$, $\alpha \in \mathcal{C}[\triangle]$, and $x \in PS$ then $\delta x.\alpha \in \mathcal{C}[\triangle]$.*

Observe that if $\mathcal{C} \subseteq Q_p$PDAG then $\mathcal{C}[\triangle] \subseteq Q_p$PDAG: closure does not question properness. We also have the following easy proposition, which makes precise the interplay between elements of $\triangle$ in the general case:

[3] Other closure principles could have been defined in a similar way, would the underlying propositional language contain other connectives.

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

**Proposition 1** *For every subset $\mathcal{C}$ of $Q$PDAG and every finite subsets $\triangle_1, \triangle_2$ of $\{\vee, \wedge, \neg, \exists, \forall\}$, we have:*

- $\mathcal{C}[\emptyset] = \mathcal{C}$.
- *If* $\triangle_1 \subseteq \triangle_2$ *then* $\mathcal{C}[\triangle_1] \subseteq \mathcal{C}[\triangle_2]$.
- $(\mathcal{C}[\triangle_1])[\triangle_2] \subseteq \mathcal{C}[\triangle_1 \cup \triangle_2]$.
- *If* $\triangle_1 \subseteq \triangle_2$ *or* $\triangle_2 \subseteq \triangle_1$ *then* $(\mathcal{C}[\triangle_1])[\triangle_2] = \mathcal{C}[\triangle_1 \cup \triangle_2]$.

Before focusing on some specific "operators", we add to succinctness the following notions of polynomial translation and polynomial equivalence, which prove helpful in the following evaluations:

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

Like $\geq_s$, $\geq_{\mathcal{P}}$ is a preorder (i.e., a reflexive and transitive relation) over the power set of $Q$PDAG. It refines the spatial efficiency preorder $\geq_s$ over $Q$PDAG in the sense that for any two subsets $\mathcal{C}_1$ and $\mathcal{C}_2$ of $Q$PDAG, if $\mathcal{C}_1 \geq_{\mathcal{P}} \mathcal{C}_2$, then $\mathcal{C}_1 \geq_s \mathcal{C}_2$ (but the converse does not hold in general). Thus, if $\mathcal{C}_1$ is polynomially translatable into $\mathcal{C}_2$, we have that $\mathcal{C}_2$ is at least as succinct as $\mathcal{C}_1$. Furthermore, whenever $\mathcal{C}_1$ is polynomially translatable into $\mathcal{C}_2$, every query which is supported in polynomial time in $\mathcal{C}_2$ also is supported in polynomial time in $\mathcal{C}_1$; and conversely, every query which is not supported in polynomial time in $\mathcal{C}_1$ unless the polynomial hierarchy collapses cannot be supported in polynomial time in $\mathcal{C}_2$, unless the polynomial hierarchy collapses.

The corresponding indifference relation $\sim_{\mathcal{P}}$ given by $\mathcal{C}_1 \sim_{\mathcal{P}} \mathcal{C}_2$ iff $\mathcal{C}_1 \geq_{\mathcal{P}} \mathcal{C}_2$ and $\mathcal{C}_2 \geq_{\mathcal{P}} \mathcal{C}_1$, is an equivalence relation; when $\mathcal{C}_1 \sim_{\mathcal{P}} \mathcal{C}_2$, $\mathcal{C}_1$ and $\mathcal{C}_2$ 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.

Before presenting some useful polynomial equivalences, we first need to introduce the notion of stability under uniform renaming. It characterizes the subsets $\mathcal{C}$ of $Q_p$PDAG for which, intuitively, the choice of variables names does not really matter; technically it allows to rename (bound) variables in a formula $\alpha$ of $\mathcal{C}$ without leaving the fragment.

**Definition 7 (stability under uniform renaming)** *Let $\mathcal{C}$ be any subset of $Q_p$PDAG. $\mathcal{C}$ is* stable under uniform renaming *iff for every $\alpha \in \mathcal{C}$, there exists arbitrarily many distinct bijections $r$ from $Var(\alpha)$ to subsets $V$ of fresh variables from $PS$ (i.e., not occurring in $\alpha$) such that the formula $r(\alpha)$ obtained by replacing in $\alpha$ (in a uniform way) every free occurrence of $x \in Var(\alpha)$ by $r(x)$ belongs to $\mathcal{C}$ as well.*

We are now ready to present more specific results:

**Proposition 2** *Let $\mathcal{C}$ be any subset of $Q_p$PDAG, s.t. $\mathcal{C}$ is stable under uniform renaming. We have:*

- $(\mathcal{C}[\exists])[\vee] \sim_{\mathcal{P}} (\mathcal{C}[\vee])[\exists] \sim_{\mathcal{P}} \mathcal{C}[\vee, \exists]$.
- $(\mathcal{C}[\forall])[\wedge] \sim_{\mathcal{P}} (\mathcal{C}[\wedge])[\forall] \sim_{\mathcal{P}} \mathcal{C}[\wedge, \forall]$.

It is important to note that such polynomial equivalences, showing in some sense that the "sequential" closure of a propositional fragment stable under uniform renaming by a set of "operators" among $\{\vee, \exists\}$ (resp. among $\{\wedge, \forall\}$) is equivalent to its "parallel" closure, cannot be systematically guaranteed for any choices of fragments and "operators". For instance, if $\mathcal{C}$ is the set $L_{PS} \cup \{\top, \bot\}$, then $(\mathcal{C}[\vee])[\wedge]$ is the set of all CNF formulas, $(\mathcal{C}[\wedge])[\vee]$ is the set of all DNF formulas, and $\mathcal{C}[\vee, \wedge]$ is the set of all PDAG-NNF formulas. From the succinctness results reported in [3], it is easy to conclude that those three fragments are not pairwise polynomially equivalent. Similarly, if $\mathcal{C}$ is the set of all clauses over $PS$, then $(\mathcal{C}[\wedge])[\exists]$ and $\mathcal{C}[\wedge, \exists]$ are polynomially equivalent to CNF$[\exists]$, but $(\mathcal{C}[\exists])[\wedge]$ is polynomially equivalent to CNF, which is not polynomially equivalent to CNF$[\exists]$ (this follows from the forthcoming Proposition 8).

## 4.2 Disjunctive Closures

In the rest of this paper, we will focus on the two disjunctive closure principles $[\vee]$ (closure by disjunction), $[\exists]$ (closure by forgetting), and their combinations. At the start, this choice was motivated by the fact that any closure $\mathcal{C}[\exists]$ obviously satisfies forgetting, which is an important transformation for a number of applications, including planning, diagnosis, reasoning about action and change, reasoning under inconsistency (see e.g. [2, 8, 9] for details), while any closure $\mathcal{C}[\vee]$ clearly preserves the crucial query **CO** and transformation **CD**.

Our purpose is now to locate on the KC map all languages obtained by applying the disjunctive closure principles to the eight languages PDAG-NNF, DNNF, CNF, OBDD$_<$ DNF, PI, IP, MODS considered (among others) in [3]; all those languages are subsets of PDAG:

- PDAG-NNF is the subset of PDAG consisting of negation normal form formulas.
- DNNF is the subset of PDAG-NNF consisting of decomposable negation normal form formulas.
- CNF is the subset of PDAG-NNF consisting of conjunctive normal form formulas.
- OBDD$_<$ is the subset of DNNF consisting of ordered binary decision diagrams. $<$ is a strict and complete ordering over $PS$ and we assume the ordered set $(PS, <)$ of order type $\eta$ (the order type of the set of rational numbers with its familiar ordering).[5]
- DNF is the subset of DNNF consisting of disjunctive normal form formulas.
- PI is the subset of CNF consisting of all prime implicates (or Blake) formulas.
- IP is the subset of DNF consisting of all prime implicants formulas.
- MODS is the subset of DNF consisting of disjunctions $\alpha$ of canonical terms over $Var(\alpha)$.[6]

For space reasons, we cannot provide formal definitions of those languages here (they can be found e.g. in [3, 12]).

It is easy to prove that the eight languages PDAG-NNF, DNNF, CNF, OBDD$_<$ DNF, PI, IP, MODS are stable under uniform renaming. Hence, thanks to Propositions 1 and 2, it is enough to consider the three fragments $\mathcal{C}[\exists]$, $\mathcal{C}[\vee]$, and $\mathcal{C}[\vee, \exists]$ for $\mathcal{C}$ being any on the eight above languages. Applying the three disjunctive closure principles $[\vee]$, $[\exists]$, and $[\vee, \exists]$ to the eight languages leads to consider twenty-four fragments. The following result shows that many fragments do not need to be considered separately, because they are polynomially equivalent.

[5] This technical, yet harmless, condition ensures that OBDD$_<$ is stable under uniform renaming, which cannot be guaranteed in the general case for this fragment (due to the constraint of compatibility with $<$ imposed to every variable path from the root of any OBDD$_<$ formula to any of its leaves).

[6] If $\alpha$ is a MODS formula and $x \in Var(\alpha)$ then every term of $\alpha$ contains a literal $l$ s.t. $var(l) = x$.

**Proposition 3**

- $\mathtt{CNF}[\exists] \sim_{\mathcal{P}} \mathtt{CNF}[\vee, \exists]$ $\sim_{\mathcal{P}} \mathtt{PDAG\text{-}NNF}[\exists] \sim_{\mathcal{P}} \mathtt{PDAG\text{-}NNF}[\vee, \exists]$.
- $\mathtt{PDAG\text{-}NNF} \sim_{\mathcal{P}} \mathtt{PDAG\text{-}NNF}[\vee]$.
- $\mathtt{DNNF} \sim_{\mathcal{P}} \mathtt{DNNF}[\vee] \sim_{\mathcal{P}} \mathtt{DNNF}[\exists] \sim_{\mathcal{P}} \mathtt{DNNF}[\vee, \exists]$.
- $\mathtt{OBDD}_<[\exists] \sim_{\mathcal{P}} \mathtt{OBDD}_<[\vee, \exists]$.
- $\mathtt{PI}[\vee] \sim_{\mathcal{P}} \mathtt{PI}[\vee, \exists]$.
- $\mathtt{PI} \sim_{\mathcal{P}} \mathtt{PI}[\exists]$.
- $\mathtt{DNF} \sim_{\mathcal{P}} \mathtt{DNF}[\vee] \sim_{\mathcal{P}} \mathtt{DNF}[\exists] \sim_{\mathcal{P}} \mathtt{DNF}[\vee, \exists] \sim_{\mathcal{P}} \mathtt{IP}[\vee] \sim_{\mathcal{P}} \mathtt{IP}[\exists]$ $\sim_{\mathcal{P}} \mathtt{IP}[\vee, \exists] \sim_{\mathcal{P}} \mathtt{MODS}[\vee] \sim_{\mathcal{P}} \mathtt{MODS}[\vee, \exists]$.
- $\mathtt{MODS} \sim_{\mathcal{P}} \mathtt{MODS}[\exists]$.

In the light of Proposition 3, it is thus enough to consider the five remaining languages, only, i.e., $\mathtt{CNF}[\exists]$, $\mathtt{CNF}[\vee]$, $\mathtt{OBDD}_<[\exists]$, $\mathtt{OBDD}_<[\vee]$, and $\mathtt{PI}[\vee]$; "remaining" means here not identified as polynomially equivalent to one of the languages already located within the KC map in [3].

## 4.3 Queries and Transformations

Let us present first the general results we obtained about tractable queries and transformations:

**Proposition 4** *Let $\mathcal{C}$ be any subset of $Q_{\mathcal{P}}\mathtt{PDAG}$.*

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

We have also derived some more specific results, about the five remaining languages:

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

| | **CO** | **VA** | **CE** | **IM** | **EQ** | **SE** | **CT** | **ME** | **MC** |
|---|---|---|---|---|---|---|---|---|---|
| $\mathtt{CNF}[\exists]$ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| $\mathtt{CNF}[\vee]$ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | √ |
| $\mathtt{OBDD}_<[\exists]$ | √ | ○ | √ | ○ | ○ | ○ | ○ | √ | √ |
| $\mathtt{OBDD}_<[\vee]$ | √ | ○ | √ | ○ | ○ | ○ | ○ | √ | √ |
| $\mathtt{PI}[\vee]$ | √ | ○ | √ | ○ | ○ | ○ | ○ | √ | √ |

**Table 1.** Subsets of the $\exists\mathtt{PDAG\text{-}NNF}$ language and their corresponding polytime queries. √ means "satisfies" and ○ means "does not satisfy unless P = NP."

This proposition shows in particular that $\mathtt{OBDD}_<[\exists]$, $\mathtt{OBDD}_<[\vee]$, $\mathtt{PI}[\vee]$ satisfy the same tractable queries (among those considered here); such queries include all the tractable queries offered by $\mathtt{CNF}[\vee]$; $\mathtt{CNF}[\exists]$ offers no tractable query.

As to transformations, we have obtained the following results:

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

| | **CD** | **FO** | **SFO** | $\wedge$**C** | $\wedge$**BC** | $\vee$**C** | $\vee$**BC** | $\neg$**C** |
|---|---|---|---|---|---|---|---|---|
| $\mathtt{CNF}[\exists]$ | √ | √ | √ | √ | √ | √ | √ | ○$*$ |
| $\mathtt{CNF}[\vee]$ | √ | ● | √ | **?** | √ | √ | √ | **?** |
| $\mathtt{OBDD}_<[\exists]$ | √ | √ | √ | ○ | √ | √ | √ | ○ |
| $\mathtt{OBDD}_<[\vee]$ | √ | **?** | √ | ○ | √ | √ | √ | ○ |
| $\mathtt{PI}[\vee]$ | √ | √ | √ | ○ | **?** | √ | √ | ○ |

**Table 2.** Subsets of the $\exists\mathtt{PDAG\text{-}NNF}$ language and their corresponding polytime transformations. √ means "satisfies," ● means "does not satisfy," ○ means "does not satisfy unless P=NP." and ○$*$ means "does not satisfy unless the polynomial hierarchy collapses."

This proposition shows in particular that $\mathtt{OBDD}_<[\exists]$ satisfies at least all the transformations offered by $\mathtt{OBDD}_<[\vee]$ and $\mathtt{PI}[\vee]$; $\mathtt{CNF}[\vee]$ does not satisfy **FO**; $\mathtt{CNF}[\exists]$ satisfies all transformations but $\neg$**C**.

Propositions 5 and 6 also show that preservation results by disjunctive closures (as the ones reported in Proposition 4 and related to **CO**, **CD**, $\wedge$**C**, $\wedge$**BC**, $\vee$**C**, $\vee$**BC**) do not hold for **VA**, **IM**, **EQ**, **SE**, **CT**, or $\neg$**C**: moving from a fragment $\mathcal{C}$ to one of its disjunctive closures $\mathcal{C}[\vee]$, $\mathcal{C}[\exists]$, or $\mathcal{C}[\vee, \exists]$ may easily lead to give up **VA**, **IM**, **EQ**, **SE**, **CT** and $\neg$**C** (just take $\mathcal{C}$ = $\mathtt{OBDD}_<$).

## 4.4 Succinctness

For space reasons, we split our succinctness results into two propositions (and two tables). In the first table, we compare w.r.t. spatial efficiency $\leq_s$ the five remaining fragments we have considered.

**Proposition 7** *The results in Table 3 hold.*

| | $\mathtt{CNF}[\exists]$ | $\mathtt{CNF}[\vee]$ | $\mathtt{OBDD}_<[\exists]$ | $\mathtt{OBDD}_<[\vee]$ | $\mathtt{PI}[\vee]$ |
|---|---|---|---|---|---|
| $\mathtt{CNF}[\exists]$ | $\sim_s$ | $\leq_s$ | $\leq_s$ | $\leq_s$ | $\leq_s$ |
| $\mathtt{CNF}[\vee]$ | $\not\leq_s$ | $\sim_s$ | $\not\leq_s$ | $\not\leq_s$ | $\leq_s$ |
| $\mathtt{OBDD}_<[\exists]$ | $\not\leq_s^*$ | $\not\leq_s^*$ | $\sim_s$ | $\leq_s$ | ? |
| $\mathtt{OBDD}_<[\vee]$ | $\not\leq_s^*$ | $\not\leq_s^*$ | ? | $\sim_s$ | ? |
| $\mathtt{PI}[\vee]$ | $\not\leq_s$ | $\not\leq_s^*$ | $\not\leq_s$ | $\not\leq_s$ | $\sim_s$ |

**Table 3.** Succinctness of target compilation languages. $*$ means that the result holds unless the polynomial hierarchy collapses.

In the second table, we compare w.r.t. $\leq_s$ the five remaining fragments with the eight fragments $\mathtt{PDAG\text{-}NNF}$, $\mathtt{DNNF}$, $\mathtt{CNF}$, $\mathtt{OBDD}_<$ $\mathtt{DNF}$, $\mathtt{PI}$, $\mathtt{IP}$, $\mathtt{MODS}$:

**Proposition 8** *The results in Table 4 hold.*

| | $\mathtt{CNF}[\exists]$ | $\mathtt{CNF}[\vee]$ | $\mathtt{OBDD}_<[\exists]$ | $\mathtt{OBDD}_<[\vee]$ | $\mathtt{PI}[\vee]$ |
|---|---|---|---|---|---|
| $\mathtt{PDAG\text{-}NNF}$ | $?, \geq_s$ | $\leq_s, \not\geq_s$ | $\leq_s, \not\geq_s^*$ | $\leq_s, \not\geq_s^*$ | $\leq_s, \not\geq_s$ |
| $\mathtt{CNF}$ | $\not\leq_s, \geq_s$ | $\not\leq_s, \geq_s$ | $\not\leq_s, \not\geq_s^*$ | $\not\leq_s, \not\geq_s^*$ | $\not\leq_s, \not\geq_s^*$ |
| $\mathtt{DNNF}$ | $\not\leq_s^*, \geq_s$ | $\not\leq_s^*, \not\geq_s$ | $\leq_s, ?$ | $\leq_s, ?$ | $?, \not\geq_s^*$ |
| $\mathtt{DNF}$ | $\not\leq_s, \geq_s$ | $\not\leq_s, \geq_s$ | $\not\leq_s, \geq_s$ | $\not\leq_s, \geq_s$ | $\not\leq_s, \geq_s$ |
| $\mathtt{OBDD}_<$ | $\not\leq_s, \geq_s$ | $\not\leq_s, \not\geq_s$ | $\not\leq_s, \geq_s$ | $\not\leq_s, \geq_s$ | $\not\leq_s, \not\geq_s$ |
| $\mathtt{PI}$ | $\not\leq_s, \geq_s$ | $\not\leq_s, \geq_s$ | $\not\leq_s, ?$ | $\not\leq_s, ?$ | $\not\leq_s, \geq_s$ |
| $\mathtt{IP}$ | $\not\leq_s, \geq_s$ | $\not\leq_s, \geq_s$ | $\not\leq_s, \geq_s$ | $\not\leq_s, \geq_s$ | $\not\leq_s, \geq_s$ |
| $\mathtt{MODS}$ | $\not\leq_s, \geq_s$ | $\not\leq_s, \geq_s$ | $\not\leq_s, \geq_s$ | $\not\leq_s, \geq_s$ | $\not\leq_s, \geq_s$ |

**Table 4.** Succinctness of target compilation languages. $*$ means that the result holds unless the polynomial hierarchy collapses.

## 5 DISCUSSION

In the previous sections, we have listed a number of general results (linking propositional fragments to their disjunctive closures) and specific results (i.e., pertaining to some specific fragments).

As to general results, Proposition 4 shows in particular that, under the stability under uniform renaming requirement (which is not very demanding), whenever a $Q_p$PDAG fragment $\mathcal{C}$ satisfies **CO** and **CD**, the associated fragment $\mathcal{C}[\vee, \exists]$ satisfies **CO**, **CD**, **CE**, **ME**, **MC**, $\vee$**C** and **FO**. Thus $\mathcal{C}[\vee, \exists]$ gives $\vee$**C** and **FO** "for free"; indeed, from the obvious inclusion $\mathcal{C} \subseteq \mathcal{C}[\vee, \exists]$, it turns out that $\mathcal{C}[\vee, \exists]$ is at least as succinct as $\mathcal{C}$. Furthermore, Proposition 4 shows that, under the same stability condition, closure by forgetting preserves $\mathbf{CO}$, $\mathbf{CD}$, $\wedge\mathbf{C}$, $\wedge\mathbf{BC}$, $\vee\mathbf{C}$, and $\vee\mathbf{BC}$; thus moving from $\mathcal{C}$ to its closure by forgetting $\mathcal{C}[\exists]$ may lead to improve the spatial efficiency (and, for sure, not to decrease it!), without a complexity shift w.r.t. any of these queries and transformations; thus, for instance, $\texttt{OBDD}_<[\exists]$ (resp. $\texttt{CNF}[\exists]$) is stricly more succinct than $\texttt{OBDD}_<$(resp. CNF).

As to specific results, our results show PI$[\vee]$ as a fragment challenging PI, when **VA**, **IM**, **EQ**, **SE** are not required by the application under consideration. Indeed, like PI, its closure PI$[\vee]$ satisfies **CO**, **CE**, and **ME**; besides, PI$[\vee]$ offers more tractable transformations than PI and is strictly more succinct than it.

Our results also show $\texttt{OBDD}_<[\exists]$ (which is polynomially equivalent to $\texttt{OBDD}_<[\vee, \exists]$) as an interesting alternative to DNNF for applications where $\wedge\mathbf{BC}$ is required (DNNF does not satisfy $\wedge\mathbf{BC}$). While we ignore whether DNNF is strictly more succinct than $\texttt{OBDD}_<[\exists]$ or not, we know that $\texttt{OBDD}_<[\exists]$ is strictly more succinct than $\texttt{OBDD}_<$ and DNF. Furthermore, we know that $\texttt{OBDD}_<[\exists]$ satisfies the same polytime queries as DNNF or DNF, and the same polytime transformations as DNF, and strictly more than DNNF.

Thus, $\texttt{OBDD}_<[\exists]$ can prove useful for applications where **CO**, **CE**, **ME**, **MC**, **CD**, **FO**, $\wedge\mathbf{BC}$, $\vee\mathbf{C}$ are enough. As a matter of example, consider a preference-based search problem (e.g. the configuration of a "simple product" like a travel) where the input data is given by some hard constraints (the feasible travels), plus some soft constraints encoding the current choices of the user. Notice that since several variables can be involved in the soft constraints, complex choices can be represented (for instance $\alpha = (loc_1 \Leftrightarrow (acc_1 \vee acc_2 \vee acc_3)) \wedge (\neg loc_2 \vee acc_4) \wedge (\neg acc_3 \vee \neg acc_4)$ expressing the user's choices as to the possible locations and the types of accommodations); this is a great advantage over current systems which restrict the representation of user's choices to literals. If the conjunction of the constraints is inconsistent, the soft constraints have to be relaxed. A way to perform this relaxation is to weaken the soft constraints by forgetting some variables in them (see [9]). Thus, $\exists acc_3.\alpha \equiv ((\neg loc_1 \vee acc_1 \vee acc_2 \vee acc_4) \wedge (\neg acc_1 \vee loc_1) \wedge (\neg acc_2 \vee loc_1) \wedge (\neg loc_2 \vee acc_4))$ is the relaxation of $\alpha$ obtained by removing whatever was imposed on the accomodation $acc_3$. Such a relaxation can prove sufficient to lead to a solution. In the light of our results, each step of such an interactive process (which consists of consistency checks, followed by relaxation steps until consistency is reached, and finally the generation of some solutions) can be achieved in polynomial time, provided that the hard constraints and the soft constraints have been first compiled into $\texttt{OBDD}_<$ formulas; the approach is as follows: (1) conjoin the hard and the soft constraints, which can be done in polynomial time since $\texttt{OBDD}_<[\exists]$ satisfies $\wedge\mathbf{BC}$, (2) determine in polynomial time whether the result is consistent or not ($\texttt{OBDD}_<[\exists]$ satisfies **CO**), (3) if this is the case generate in polynomial time one or several solutions ($\texttt{OBDD}_<[\exists]$ satisfies **ME**) else forget in polynomial time some variables in the soft constraints ($\texttt{OBDD}_<[\exists]$ satisfies **FO**) and resume from (1).

## 6 CONCLUSION

In this paper, we have extended the KC map with new propositional fragments obtained by applying disjunctive closure principles to several fragments studied so far. We have investigated two closure principles, disjunction and implicit forgetting (i.e., existential quantification), and their combinations.

This paper calls for a number of perspectives. One of them consists in removing the question marks which remain in the previous tables. Another issue for further research concerns the principle of closure by disjunction: notwithstanding the fact that it "commutes" with forgetting, it has its own interest since it allows to render complete every incomplete propositional fragments containing the set of all terms over $PS$.[7] Accordingly, another perspective consists in extending further the KC map by applying the disjunctive closures at work here to other propositional languages (e.g. the set of all Horn CNF formulas), and evaluating the resulting fragments. [6] is a first step in this direction.

## ACKNOWLEDGEMENTS

The authors would like to thank the anonymous reviewers for their helpful comments. They have been partly supported by the ANR project PHAC (ANR-05-BLAN-0384). Pierre Marquis has also been partly supported by the IUT de Lens and the Région Nord/Pas-de-Calais.

## REFERENCES

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

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

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

[4] R. Dechter and I. Rish, 'Directional resolution: the Davis-Putnam procedure, revisited', in *Proceedings of KR'94*, pp. 134–145, (1994).

[5] H. Fargier and P. Marquis, 'On the use of partially ordered decision graphs in knowledge compilation and quantified Boolean formulae', in *Proceedings of AAAI'06*, pp. 42–47, (2006).

[6] H. Fargier and P. Marquis, 'Extending the knowledge compilation map: Krom, Horn, affine and beyond', in *Proceedings of AAAI'08*, (2008). To appear.

[7] G. Gogic, H.A. Kautz, Ch.H. Papadimitriou, and B. Selman, 'The comparative linguistics of knowledge representation', in *Proceedings of IJCAI'95*, pp. 862–869, (1995).

[8] J. Lang, P. Liberatore, and P. Marquis, 'Propositional independence - formula-variable independence and forgetting', *Journal of Artificial Intelligence Research*, **18**, 391–443, (2003).

[9] J. Lang and P. Marquis, 'Resolving inconsistencies by variable forgetting', in *Proceedings of KR'02*, pp. 239–250, (2002).

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

[11] S. Subbarayan, L. Bordeaux, and Y. Hamadi, 'Knowledge compilation properties of tree-of-BDDs', in *Proceedings of AAAI'07*, pp. 502–507, (2007).

[12] M. Wachter and R. Haenni, 'Propositional DAGs: A new graph-based language for representing Boolean functions', in *Proceedings of KR'06*, pp. 277–285, (2006).

[7] Indeed, if $\mathcal{C}$ consists of all terms over $PS$ – obviously, an incomplete fragment in the sense that some propositional formulas like $a \vee b$ are not equivalent to any term – then $\mathcal{C}[\vee]$ is equal to the complete fragment DNF.