Extending the Knowledge Compilation Map: Closure Principles

Hélène Fargier and Pierre Marquis

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 dozens 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 ∧, ∨, and ¬ (∧C, ∧BC, ∨C, ∨BC, ¬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 (∨) and implicit forgetting (¬), and their combinations. Roughly speaking, the disjunction principle when applied to a fragment C leads to a fragment C[∨] which allows disjunctions of formulas from C. While implicit forgetting applied to a fragment C leads to a fragment C[¬] which allows existentially quantified formulas from C. Obviously enough, whatever C, C[∨] satisfies polytime closure under ∨ (∨C) and C[¬] 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 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 C satisfies CO (resp. CD), then C[∨] and C[¬] 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 QPDAG:

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

- QPDAG is the set of all finite, single-rooted DAGs α (called formulas) where each leaf node is labeled by a literal over PS or one of the two Boolean constants ⊤ or ⊥, and each internal node is labeled by ∧ or ∨ and has arbitrarily many children or is labeled by ¬, ∃x or ∀x (where x ∈ PS) and has just one child.
- Qα,P S PDAG is the subset of all proper formulas of QPDAG, where a formula α is proper iff for every literal l = x or l = ¬x labelling a leaf of α, at most one path from the root of α to this leaf contains quantifications of the form ∃x or ∀x, and if such a path exists, it is the unique path from the root of α to the leaf.

Restricting the language QPDAG to proper formulas α ensures that every occurrence of a variable x corresponding to a literal at a leaf of α 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 $Q_pPDAG$ obtained by removing the possibility to have internal nodes labeled by $3$ or $v$; PDAG−NNF [3] (resp. PDAG−NNF, resp. PDAG−NNF) is the subset of $Q_pPDAG$ obtained by removing the possibility to have internal nodes labeled by $\neg, 3$ or $v$ (resp. $\neg, v, \neg, 3$). Distinguished formulas from $QPDAG$ are the literals over $PS$; if $V$ is any subset of $PS$, $LV$ denotes the set of all literals built over $V$, i.e., $\{ x, \neg x \mid x \in V \}$. If a literal $l$ of $L_PV$ 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 from a list of atoms $x$, we have $var(l) = x$. A clause (resp. a term) is a (finite) disjunction (resp. conjunction) of literals or the constant $\bot$. The size of a formula is defined inductively in the standard way; otherwise it has the form $\exists \alpha$ or the constant $\bot$. A clause (resp. a term) is a (finite) disjunction (resp. conjunction) of literals or the constant $\bot$. The size of a formula is defined inductively in the standard way; otherwise it has the form $\exists \alpha$. A clause (resp. a term) is a (finite) disjunction (resp. conjunction) of literals or the constant $\bot$. The size of a formula is defined inductively in the standard way; otherwise it has the form $\exists \alpha$. 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 recapping it here; we extend them to $Q_pPDAG$ 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 $Q_pPDAG$ formulas.

Definition 2 (queries) Let $C$ denote any subset of $Q_pPDAG$.

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

Definition 3 (transformations) Let $C$ denote any subset of $Q_pPDAG$.

- $C$ satisfies $CD$ iff there exists a polytime algorithm that maps every formula $\alpha$ from $C$ and every consistent term $\gamma$ to a formula from $C$ 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 $\gamma$ (resp. $\bot$) if $x$ (resp. $\neg x$) is a positive (resp. negative) literal of $\gamma$.
- $C$ satisfies $FO$ iff there exists a polytime algorithm that maps every formula $\alpha$ from $C$ and every subset $X$ of variables from $PS$ to a formula from $C$ equivalent to $\exists X. \alpha$. If the property holds for each singleton $X$, we say that $C$ satisfies $SFO$.
- $C$ 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 $C$ to a formula of $C$ that is logically equivalent to $\alpha_1 \wedge \ldots \wedge \alpha_n$ (resp. $\alpha_1 \vee \ldots \vee \alpha_n$).
- $C$ satisfies $\forall C$ (resp. $\exists C$) iff there exists a polytime algorithm that maps every pair of formulas $\alpha$ and $\beta$ from $C$ to a formula of $C$ that is logically equivalent to $\alpha \wedge \beta$ (resp. $\alpha \vee \beta$).
- $C$ satisfies $\neg C$ iff there exists a polytime algorithm that maps every formula $\alpha$ from $C$ to a formula of $C$ 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 $QPDAG$ formulas:

Definition 4 (succinctness) Let $C_1$ and $C_2$ be two subsets of $PDAG$. $C_1$ is at least as succinct as $C_2$, denoted $C_1 \preceq C_2$, iff there exists a polynomial $p$ such that for every formula $\alpha \in C_2$, there exists an equivalent formula $\beta \in C_1$ where $|\beta| \leq p(|\alpha|)$.

$\sim_s$ is the symmetric part of $\preceq_s$ defined by $C_1 \sim_s C_2$ iff $C_1 \preceq_s C_2$ and $C_2 \preceq_s C_1$. $\preceq_s$ is the asymmetric part of $\preceq_s$ defined by $C_1 \prec_s C_2$ iff $C_1 \preceq_s C_2$ and $C_2 \npreceq_s C_1$.

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 $C$ be a subset of $QPDAG$ and $\Delta$ be any finite subset of $\{ \{v, \wedge, \forall, \exists, v\} \}$. $C[\Delta]$ is the subset of $QPDAG$ inductively defined as follows:\(^4\)

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

Observe that if $C \subseteq Q_pPDAG$ then $C[\Delta] \subseteq Q_pPDAG$; closure does not question properness. We also have the following easy proposition, which makes precise the interplay between elements of $\Delta$ 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 $\Delta = \{ \delta_1, \ldots, \delta_n \}$, we shall write $C[\delta_1, \ldots, \delta_n]$ instead of $C[\{ \delta_1, \ldots, \delta_n \}]$. 

2
Proposition 1 For every subset \( C \) of \( Q_{PDAG} \) and every finite subsets \( \Delta_1, \Delta_2 \) of \( \{ V, \land, \lor, \exists, \forall \} \), we have:

- \( \{ \} = C\).
- If \( \Delta_1 \subseteq \Delta_2 \) then \( C[\Delta_1] \subseteq C[\Delta_2] \).
- \( (C[\Delta_1])[\Delta_2] \subseteq C[\Delta_1 \cup \Delta_2] \).
- If \( \Delta_1 \subseteq \Delta_2 \) or \( \Delta_2 \subseteq \Delta_1 \) then \( C[\Delta_1][\Delta_2] = C[\Delta_1 \cup \Delta_2] \).

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 \( C_1 \) and \( C_2 \) be two subsets of \( Q_{PDAG} \). \( C_1 \) is said to be polynomially translatable into \( C_2 \), noted \( C_1 \geq_{\mathcal{P}} C_2 \), if there exists a polytime algorithm \( f \) such that for every \( \alpha \in C_1 \), we have \( f(\alpha) \in C_2 \) and \( f(\alpha) \equiv \alpha \).

Like \( \geq_{\mathcal{P}} \), \( \geq_{\mathcal{P}} \) is a preorder (i.e., a reflexive and transitive relation) over the power set of \( Q_{PDAG} \). It refines the spatial efficiency preorder \( \geq_{\mathcal{P}} \) over \( Q_{PDAG} \) in the sense that for any two subsets \( C_1 \) and \( C_2 \) of \( Q_{PDAG} \), if \( C_1 \geq_{\mathcal{P}} C_2 \), then \( C_1 \geq C_2 \) (but the converse does not hold in general). Thus, if \( C_1 \) is polynomially translatable into \( C_2 \), we have that \( C_2 \) is at least as succinct as \( C_1 \). Furthermore, whenever \( C_1 \) is polynomially translatable into \( C_2 \), every query which is supported in polynomial time in \( C_1 \) is also supported in polynomial time in \( C_2 \); and conversely, every query which is not supported in polynomial time in \( C_1 \) unless the polynomial hierarchy collapses cannot be supported in polynomial time in \( C_2 \), unless the polynomial hierarchy collapses.

The corresponding indifference relation \( \sim_{\mathcal{P}} \) given by \( C_1 \sim_{\mathcal{P}} C_2 \) iff \( C_1 \geq_{\mathcal{P}} C_2 \) and \( C_2 \geq_{\mathcal{P}} C_1 \), is an equivalence relation; when \( C_1 \) is polynomially translatable into \( C_2 \), every query which is supported in polynomial time in \( C_2 \) also is supported in polynomial time in \( C_1 \); and conversely, every query which is not supported is polynomially translatable into \( C_1 \) unless the polynomial hierarchy collapses cannot be supported in polynomial time in \( C_2 \), unless the polynomial hierarchy collapses.

Definition 7 (stability under uniform renaming) Let \( C \) be any subset of \( Q_{PDAG} \). \( C \) is stable under uniform renaming if for every \( \alpha \in C \), there exists arbitrarily many bijections \( r \) from \( V \) to subsets \( V \) of fresh variables from \( P \) (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 \( x \in V \) by \( r(x) \) belongs to \( C \) as well.

We are now ready to present more specific results:

Proposition 2 Let \( C \) be any subset of \( Q_{PDAG} \), s.t. \( C \) is stable under uniform renaming. We have:

- \( (\{3\})[V] \sim_{\mathcal{P}} (\{V\})[\{3\}] \sim_{\mathcal{P}} C[\{V, 3\}] \).
- \( (\{V\})[\alpha] \sim_{\mathcal{P}} (\{\alpha\})[\{V\}] \sim_{\mathcal{P}} C[\{\alpha, V\}] \).

It is important to note that some polynomial equivalences, showing in some sense that “sequential” closure of a propositional fragment stable under uniform renaming by a set of “operators” among \( \{V, 3\} \) (resp. among \( \{\land, V\} \)) is equivalent to its “parallel” closure, cannot be systematically guaranteed for any choices of fragments and “operators”. For instance, if \( C \) is the set \( LFS \cup \{T, \bot\} \), then \( C[V]\{\alpha\}[V] \) is the set of all \( CNF \) formulas, \( C[V]\{\alpha\}[\{V\}] \) is the set of all \( DNF \) formulas, and \( C[V, \exists] \) 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 \( C \) is the set of all clauses over \( PS \), then \( C[\{\alpha\}] \) and \( C[\{\exists, \exists\}] \) are polynomially equivalent to \( CNF[3] \), but \( C[\{3\}] \) is polynomially equivalent to \( CNF \), which is not polynomially equivalent to \( CNF[3] \) (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 \( \{V\} \) (closure by disjunction), \( \{3\} \) (closure by forgetting), and their combinations. At the start, this choice was motivated by the fact that any closure \( C[3] \) 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 \( C[V] \) 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 \( PDAG-NNF \), \( DNNF \), \( CNF \), \( OBDD_\leq \), \( DNF \), \( PI \), \( IP \), \( MODS \) (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.
- \( OBDD_\leq \) 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).
- \( DNF \) is the subset of \( DNNF \) consisting of disjunctive normal form formulas.
- \( PI \) is the subset of \( CNF \) consisting of all prime implicants (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 \( Var(\alpha) \).

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 \( PDAG-NNF \), \( DNNF \), \( CNF \), \( OBDD_\leq \), \( DNF \), \( PI \), \( IP \), \( MODS \) are stable under uniform renaming. Hence, thanks to Propositions 1 and 2, it is enough to consider the three fragments \( C[3] \), \( C[V] \), and \( C[V, \exists] \) for \( C \) being any on the eight above languages. Applying the three disjunctive closure principles \( \{V\}, \{3\}, \) and \( \{V, 3\} \) 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.

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

\[^{6}\text{If } \alpha \text{ is a } MODS \text{ formula and } x \in Var(\alpha) \text{ then every term of } \alpha \text{ contains a literal } l \text{ s.t. } Var(l) = x.\]
Proposition 3

- CNF[3] ∼ π CNF[v, 3]
- PDAG-NNF ∼ π PDAG-NNF[v].
- OBDD_c[3] ∼ π OBDD_c[v, 3].
- PI[v] ∼ π PI[v, 3].
- PI ∼ π PI[3].
  ∼ π IP[v, 3] ∼ π MODS[v] ∼ π MODS[v, 3].
- MODS ∼ π MODS[3].

In the light of Proposition 3, it is thus enough to consider the five remaining languages, only, i.e., CNF[3], CNF[v], OBDD_c[3], OBDD_c[v], and PI[v]; “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 C be any subset of Q_p PDAG.

- If C satisfies CO (resp. CD) then C[v], C[3] and C[v, 3] satisfy CO (resp. CD).
- If C satisfies CO and CD then C satisfies CE and ME.
- If C satisfies CO and CD then C, C[v], C[3] and C[v, 3] satisfy MC.
- If C satisfies √C (resp. √BC, √C, √BC) and is stable under uniform renaming, then C[3] satisfies √C (resp. √BC, √C, √BC).

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

Proposition 5 The results in Table 1 hold.

<table>
<thead>
<tr>
<th>CO</th>
<th>VA</th>
<th>CE</th>
<th>IM</th>
<th>EQ</th>
<th>SE</th>
<th>CT</th>
<th>ME</th>
<th>MC</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNF[3]</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>CNF[v]</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>OBDD_c[3]</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>OBDD_c[v]</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>PI[v]</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
</tr>
</tbody>
</table>

Table 1. Subsets of the ∃PDAG-NNF language and their corresponding polytime transformations. √ means “satisfies” and 0 means “does not satisfy unless P = NP.”

This proposition shows in particular that OBDD_c[3], OBDD_c[v], PI[v] satisfy the same tractable queries (among those considered here); such queries include all the tractable queries offered by CNF[v]; CNF[3] offers no tractable query.

As to transformations, we have obtained the following results:

Proposition 6 The results in Table 2 hold.

<table>
<thead>
<tr>
<th>CD</th>
<th>FO</th>
<th>SFO</th>
<th>√C</th>
<th>√BC</th>
<th>√C</th>
<th>√BC</th>
<th>¬C</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNF[3]</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>0*</td>
</tr>
<tr>
<td>OBDD_c[3]</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>0</td>
<td>√</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>OBDD_c[v]</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>0</td>
<td>√</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>PI[v]</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>0</td>
<td>√</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 2. Subsets of the ∃PDAG-NNF language and their corresponding polytime transformations. √ means “satisfies,” * means “does not satisfy,” o means “does not satisfy unless P = NP,” and o* means “does not satisfy unless the polynomial hierarchy collapses.”

This proposition shows in particular that OBDD_c[3] satisfies at least all the transformations offered by OBDD_c[v] and PI[v]; CNF[v] does not satisfy FO; CNF[3] satisfies all transformations but ¬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, ∧C, ∧BC, √C, √BC) do not hold for VA, IM, EQ, SE, CT, or ¬C: moving from a fragment C to one of its disjunctive closures C[v], C[3], or C[v, 3] may easily lead to give up VA, IM, EQ, SE, CT and ¬C (just take C = OBDD_c).

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. flat efficiency ≤_s the five remaining fragments we have considered.

Proposition 7 The results in Table 3 hold.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
</tbody>
</table>

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. ≤_s the five remaining fragments with the eight fragments PDAG-NNF, DNNF, CNF, OBDD_c, DNF, PI, IM, MODS:

Proposition 8 The results in Table 4 hold.

<table>
<thead>
<tr>
<th>PDAG-NNF</th>
<th>CNF</th>
<th>DNNF</th>
<th>OBDD_c</th>
<th>PI</th>
<th>MODS</th>
</tr>
</thead>
<tbody>
<tr>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td></td>
</tr>
</tbody>
</table>

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 \(Q_p^{PDA}\) fragment \(C\) satisfies \(CO\) and \(CD\), the associated fragment \(C[\forall, 3]\) satisfies \(CO, CD, CE, ME, MC, vC\) and \(FO\). Thus \(C[\forall, 3]\) gives \(VC\) and \(FO\) “for free”; indeed, from the obvious inclusion \(C \subseteq C[\forall, 3]\), it turns out that \(C[\forall, 3]\) is at least as succinct as \(C\). Furthermore, Proposition 4 shows that, under the same stability condition, closure by forgetting preserves \(CO, CD, \land, AC, \land BC, vC,\) and \(vBC\); thus moving from \(C\) to its closure by forgetting \(C[3]\) 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, \(OBDD_\forall[3]\) (resp. \(CNF[3]\)) is strictly more succinct than \(OBDD_\forall\) (resp. \(CNF\)).

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

Our results also show \(OBDD_\forall[3]\) (which is polynomially equivalent to \(OBDD_\forall[\forall, 3]\)) as an interesting alternative to \(DNF\) for applications where \(\land BC\) is required (\(DNF\) does not satisfy \(\land BC\)). While we ignore whether \(DNF\) is strictly more succinct than \(OBDD_\forall[3]\) or not, we know that \(OBDD_\forall[3]\) is strictly more succinct than \(OBDD_\forall\) and \(DNF\). Furthermore, we know that \(OBDD_\forall[3]\) satisfies the same polytime queries as \(DNF\) or \(DNF\), and the same polytime transformations as \(DNF\), and strictly more than \(DNF\).

Thus, \(OBDD_\forall[3]\) can prove useful for applications where \(CO, CE, ME, MC, CD, FO, \land BC, vC\) 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 \(\alpha = (loc_1 \leftrightarrow (acc_1 \lor acc_2 \lor acc_3)) \land \neg(loc_1 \lor acc_1) \land \neg(acc_1 \lor acc_2) \land \neg(acc_2 \lor loc_1) \land \neg(loc_2 \lor acc_2))\) 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, \(\exists acc_1, \alpha \equiv \neg(loc_1 \lor acc_1) \lor acc_2 \lor acc_3) \land \neg(acc_1 \lor loc_1) \land \neg(acc_2 \lor loc_1) \land \neg(loc_2 \lor acc_2))\) is the relaxation of \(\alpha\) obtained by removing whatever was imposed on the accommodation \(acc_3\). 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 \(OBDD_\forall\) formulas; the approach is as follows: (1) conjoin the hard and the soft constraints, which can be done in polynomial time since \(OBDD_\forall[3]\) satisfies \(\land BC\), (2) determine in polynomial time whether the result is consistent or not (\(OBDD_\forall[3]\) satisfies \(CO\), (3) if this is the case generate in polynomial time one or several solutions (\(OBDD_\forall[3]\) satisfies \(ME\)) else forget in polynomial time some variables in the soft constraints (\(OBDD_\forall[3]\) 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 \(P[S]\). 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