Knowledge Compilation Properties of Trees-of-BDDs, Revisited*

Hélène Fargier
IRIT-CNRS, Toulouse, France
fargier@irit.fr

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

Abstract

Recent results have shown the interest of trees-of-BDDs [Subbarayan *et al.*, 2007] as a suitable target language for propositional knowledge compilation from the practical side. In the present paper, the concept of tree-of-BDDs is extended to additional classes of data structures `C` thus leading to trees-of-`C` representations (`ToC`). We provide a number of generic results enabling one to determine the queries/transformations satisfied by `ToC` depending on those satisfied by `C`. We also present some results about the spatial efficiency of the `ToC` languages. Focusing on the $\texttt{ToOBDD}_<$ language (and other related languages), we address a number of issues that remained open in [Subbarayan *et al.*, 2007]. Among other results, we prove that $\texttt{ToOBDD}_<$ is not comparable w.r.t. succinctness with any of `CNF`, `DNF`, `DNNF` unless the polynomial hierarchy collapses. This contributes to the explanation of some empirical results reported in [Subbarayan *et al.*, 2007].

1 Introduction

This paper is concerned with "knowledge compilation" (KC), a family of approaches proposed so far for addressing the intractability of a number of AI problems. The key idea underlying KC is to pre-process parts of the available information (i.e., turning them into a compiled form) for improving online computational efficiency (see among others [Darwiche, 2001; Cadoli and Donini, 1998; Selman and Kautz, 1996; del Val, 1994]).

A important research line in KC [Gogic *et al.*, 1995; Darwiche and Marquis, 2002] addresses the following issue: *How to choose a target language for knowledge compilation?* In [Darwiche and Marquis, 2002], the authors argue that the choice of a target language must be based both on the set of queries and transformations which can be achieved in polynomial time when the data are represented in the language, as well as the spatial efficiency of the language. They pointed out a KC map which can be viewed as a multi-criteria evaluation of a number of propositional fragments, including `DNNF`, `d-DNNF`, `CNF`, `DNF`, $\texttt{OBDD}_<$, `OBDD` (the union of all $\texttt{OBDD}_<$ when $<$ varies), etc. (see [Darwiche and Marquis, 2002] for details). From there, other propositional fragments have been considered so far and put in the KC map, see for instance [Wachter and Haenni, 2006; Fargier and Marquis, 2006; Subbarayan *et al.*, 2007; Pipatsrisawat and Darwiche, 2008; Fargier and Marquis, 2008a; 2008b].

Recent experimental results have shown the practical interest of trees-of-BDDs [Subbarayan *et al.*, 2007] as a target language for propositional knowledge compilation: it turns out that the tree-of-BDDs language renders feasible the compilation of a number of benchmarks which cannot be compiled into `d-DNNF` due to space limitations.

In the present paper, we elaborate on the tree-of-BDDs language. After some formal preliminaries (Section 2), we generalize the tree-of-BDDs language to the family of `ToC` representations where `C` is any complete propositional language (Section 3). We provide a number of generic results enabling one to determine the queries/transformations satisfied by `ToC` depending on the queries/transformations satisfied by `C`. We also present some results about the spatial efficiency of the `ToC` languages. Focusing on $\texttt{ToOBDD}_<$ and some related languages, we then address a number of issues that remained open in [Subbarayan *et al.*, 2007] (Section 4): beyond **CO** and **VA**, the $\texttt{ToOBDD}_<$ language satisfies **IM** and **ME** but does not satisfy any query among **CE**, **SE** unless P = NP. Under similar assumptions from complexity theory, we demonstrate that $\texttt{ToOBDD}_<$ does not satisfy any transformation among **CD**, **FO**, $\wedge$**BC**, $\vee$**C** or $\neg$**C**. Among other succinctness results, we prove that the $\texttt{ToOBDD}_<$ language is not comparable w.r.t. succinctness with any of `CNF`, `DNF` or `DNNF` unless the polynomial hierarchy PH collapses. This contributes to the explanation of some empirical results reported in [Subbarayan *et al.*, 2007]. We conclude the paper by a discussion of the results and some perspectives (Section 5).

2 Representations and the KC Map

Trees-of-BDDs and their forthcoming generalization are not *stricto sensu* formulae. Hence we need to extend the notions of queries, transformations and succinctness at work in the KC map to such representations. Roughly speaking, a propositional representation language is a way to represent Boolean functions. Such a representation language often

*The authors They have been partly supported by the ANR project PHAC (ANR-05-BLAN-0384).

takes the form of a standard propositional language but other data structures can be used as well (e.g. Karnaugh maps, truth tables, various graphs including those from the $\texttt{OBDD}_<$ language, ... and of course trees-of-BDDs) for the representation purpose.

Formally, given a finite set of propositional variables PS, we consider Boolean functions from $\{0,1\}^X$ to $\{0,1\}$, where $X \subseteq PS$. $Var(f) = X$ is called the scope of f. The support $\Omega(f)$ of f is the set of all assignments ω of $Var(f)$ to Boolean values such that $f(\omega) = 1$. For any $X \subseteq PS$, we note by $\overline{X}$ the set $PS \setminus X$. The set of Boolean functions is equipped with the three standard internal laws, $\wedge$, $\vee$ and $\neg$. Given $X \subseteq PS$ we note by $\exists X.f$ the Boolean function with scope $Var(f) \setminus X$ that maps 1 to an assignment $\omega_{Var(f)\setminus X}$ of $Var(f) \setminus X$ iff there exists an assignment ω_X of X such that $f(\omega_{Var(f)\setminus X}.\omega_X) = 1$.

Definition 1 (representation language) *(inspired from [Gogic* et al.*, 1995]) A* (propositional) representation language *over a finite set of propositional variables PS is a set $\texttt{C}$ of data structures α (also referred to as $\texttt{C}$ representations) together with a scope function $Var : \texttt{C} \rightarrow 2^X$ with $X \subseteq PS$ and an interpretation function I which associates to each $\texttt{C}$ representation α a Boolean function $I(\alpha)$ the scope of which is $Var(\alpha)$. $\texttt{C}$ is also equipped with a size function from $\texttt{C}$ to $\mathbb{N}$ that provides the size $|\alpha|$ of any $\texttt{C}$ representation α.*[1]

Definition 2 (complete language) *A propositional representation language $\texttt{C}$ is said to be* complete *iff for any Boolean function f with $Var(f) \subseteq PS$, there exists a $\texttt{C}$ representation α such that $I(\alpha) = f$.*

Clearly enough, formulae (viewed as words) from a standard propositional language are representations of Boolean functions. The size of such a formula is the number of symbols in it. Slightly abusing words, when Σ is a propositional formula representing a Boolean function g one often says that a representation α of g is a representation of Σ instead of α is a representation of the semantics of Σ.

The $\texttt{DAG-NNF}$ language [Darwiche and Marquis, 2002] is also a complete graph-based representation language of Boolean functions. Distinguished formulae from $\texttt{DAG-NNF}$ are the literals over PS, the clauses (a clause is a finite disjunction of literals or the Boolean constant $\bot$) and the terms (a term is a finite conjunction of literals or the Boolean constant $\top$). We assume the reader to be familiar with the $\texttt{DAG-NNF}$ fragments $\texttt{DNNF}$, $\texttt{d-DNNF}$, $\texttt{CNF}$, $\texttt{DNF}$, $\texttt{FBDD}$, $\texttt{OBDD}_<$, $\texttt{OBDD}$, $\texttt{MODS}$, etc.

Obviously, all the logical notions pertaining to formulae viewed up to logical equivalence can be easily extended to any representation language $\texttt{C}$ of Boolean functions. For instance, an asssignment ω of $Var(\alpha)$ to Boolean values is said to be a *model* of a $\texttt{C}$ representation α over $Var(\alpha)$ iff $I(\alpha)(\omega) = 1$. Similarly, two representations α and β (possibly from different representation formalisms) are said to be *equivalent*, noted $\alpha \equiv \beta$, when they represent the same Boolean function. A $\texttt{C}$ representation α is *consistent* (resp. *valid*) iff α does not represent the Boolean function 0 (resp. represents the Boolean function 1). α is a *logical consequence* of β, noted $\beta \models \alpha$, iff $\Omega(I(\beta)) \subseteq \Omega(I(\alpha))$.

We are now ready to extend the notions of queries, transformations and succinctness considered in the KC map to any propositional representation language. Their importance is discussed in depth in [Darwiche and Marquis, 2002], so we refrain from recalling it here.

Definition 3 (queries) *Let $\texttt{C}$ denote a propositional representation language.*

- *$\texttt{C}$ satisfies* **CO** *(resp.* **VA***) iff there exists a polytime algorithm that maps every $\texttt{C}$ representation α to 1 if α is consistent (resp. valid), and to 0 otherwise.*
- *$\texttt{C}$ satisfies* **CE** *iff there exists a polytime algorithm that maps every $\texttt{C}$ representation α and every clause δ to 1 if $\alpha \models \delta$ holds, and to 0 otherwise.*
- *$\texttt{C}$ satisfies* **EQ** *(resp.* **SE***) iff there exists a polytime algorithm that maps every pair of $\texttt{C}$ representations α, β to 1 if $\alpha \equiv \beta$ (resp. $\alpha \models \beta$) holds, and to 0 otherwise.*
- *$\texttt{C}$ satisfies* **IM** *iff there exists a polytime algorithm that maps every $\texttt{C}$ representation α and every term γ to 1 if $\gamma \models \alpha$ holds, and to 0 otherwise.*
- *$\texttt{C}$ satisfies* **CT** *iff there exists a polytime algorithm that maps every $\texttt{C}$ representation α to a nonnegative integer that represents the number of models of α over $Var(\alpha)$ (in binary notation).*
- *$\texttt{C}$ satisfies* **ME** *iff there exists a polynomial $p(.,.)$ and an algorithm that outputs all models of an arbitrary $\texttt{C}$ representation α in time $p(|\alpha|, m)$, where m is the number of its models (over $Var(\alpha)$).*

Definition 4 (transformations) *Let $\texttt{C}$ denote a propositional representation language.*

- *$\texttt{C}$ satisfies* **CD** *iff there exists a polytime algorithm that maps every $\texttt{C}$ representation α and every consistent term γ to a $\texttt{C}$ representation β of the restriction of $I(\alpha)$ to $I(\gamma)$, i.e., $Var(\beta) = Var(\alpha) \setminus Var(\gamma)$ and $I(\beta) = \exists Var(\gamma).(I(\alpha) \wedge I(\gamma))$.*
- *$\texttt{C}$ satisfies* **FO** *iff there exists a polytime algorithm that maps every $\texttt{C}$ representation α and every subset X of variables from PS to a $\texttt{C}$ representation of $\exists X.I(\alpha)$. If the property holds for each singleton X, we say that $\texttt{C}$ satisfies* **SFO**.
- *$\texttt{C}$ satisfies* $\wedge$**C** *(resp.* $\vee$**C***) iff there exists a polytime algorithm that maps every finite set of $\texttt{C}$ representations $\alpha_1, \ldots, \alpha_n$ to a $\texttt{C}$ representation of $I(\alpha_1) \wedge \cdots \wedge I(\alpha_n)$ (resp. $I(\alpha_1) \vee \cdots \vee I(\alpha_n)$).*
- *$\texttt{C}$ satisfies* $\wedge$**BC** *(resp.* $\vee$**BC***) iff there exists a polytime algorithm that maps every pair of $\texttt{C}$ representations α and β to a $\texttt{C}$ representation of $I(\alpha) \wedge I(\beta)$ (resp. $I(\alpha) \vee I(\beta)$).*
- *$\texttt{C}$ satisfies* $\neg$**C** *iff there exists a polytime algorithm that maps every $\texttt{C}$ representation α to a $\texttt{C}$ representation of $\neg I(\alpha)$.*

[1] Of course, I refers to the interpretation function associated to the $\texttt{C}$ language, so that $I_{\texttt{C}}$ would be a more correct notation for it; nevertheless, in order to keep the notations light and since no ambiguity is possible, we refrained from indexing the functions I (as well as Var and the size function) by the associated representation language.

Definition 5 (succinctness) *Let* $\mathtt{C}_1$ *and* $\mathtt{C}_2$ *be two representation languages.* $\mathtt{C}_1$ *is at least as succinct as* $\mathtt{C}_2$*, noted* $\mathtt{C}_1 \leq_s \mathtt{C}_2$*, iff there exists a polynomial* p *such that for every* $\mathtt{C}_2$ *representation* α *there exists an equivalent* $\mathtt{C}_1$ *representation* β *where* $|\beta| \leq p(|\alpha|)$.

$\sim_s$ is the symmetric part of $\leq_s$ defined by $\mathtt{C}_1 \sim_s \mathtt{C}_2$ iff $\mathtt{C}_1 \leq_s \mathtt{C}_2$ and $\mathtt{C}_2 \leq_s \mathtt{C}_1$. $<_s$ is the asymmetric part of $\leq_s$ defined by $\mathtt{C}_1 <_s \mathtt{C}_2$ iff $\mathtt{C}_1 \leq_s \mathtt{C}_2$ and $\mathtt{C}_2 \not\leq_s \mathtt{C}_1$. Finally, $\mathtt{C}_1 \leq_s^* \mathtt{C}_2$ (resp. $\mathtt{C}_1 <_s^* \mathtt{C}_2$) means that $\mathtt{C}_1 \leq_s \mathtt{C}_2$ (resp. $\mathtt{C}_1 <_s \mathtt{C}_2$) unless the polynomial hierarchy PH collapses (which is considered very unlikely in complexity theory).

We also consider the following restriction of the succinctness relation:

Definition 6 (polynomial translation) *Let* $\mathtt{C}_1$ *and* $\mathtt{C}_2$ *be two representation languages.* $\mathtt{C}_1$ *is* polynomially translatable *into* $\mathtt{C}_2$*, noted* $\mathtt{C}_1 \geq_{\mathcal{P}} \mathtt{C}_2$*, iff there exists a polytime algorithm* A *such that for every* $\mathtt{C}_1$ *representation* α $A(\alpha)$ *is a* $\mathtt{C}_2$ *representation such that* $A(\alpha) \equiv \alpha$.

Like $\geq_s$, $\geq_{\mathcal{P}}$ is a preorder (i.e., a reflexive and transitive relation) over propositional representation languages. It refines the spatial efficiency preorder $\geq_s$ in the sense that for any $\mathtt{C}_1$ and $\mathtt{C}_2$, if $\mathtt{C}_1 \geq_{\mathcal{P}} \mathtt{C}_2$, then $\mathtt{C}_1 \geq_s \mathtt{C}_2$ (but the converse does not hold in general). We note by $\sim_{\mathcal{P}}$ the symmetric part of $\leq_{\mathcal{P}}$.

3 The `ToC` Languages

We start with the definition of trees-of-BDDs as given in [Subbarayan *et al.*, 2007] (modulo the notations used):

Definition 7 (tree-of-BDDs)

- *A* decomposition tree *of a* `CNF` *formula* Σ *is a (finite) labelled tree* T *whose set of nodes is* N*. Each node* $n \in N$ *is labelled with* $Var(n)$*, a subset of* $Var(\Sigma)$*. For each* $n \in N$*, let* $clauses(n) = \{$*clause* δ *of* Σ *s.t* $Var(\delta) \subseteq Var(n)\}$*;* T *satisfies two conditions: for every clause* δ *of* Σ *there exists* $n \in N$ *such that* $\delta \in clauses(n)$*, and for every* $x \in Var(\Sigma)$*,* $\{n \in N \mid x \in Var(n)\}$ *forms a connected subtree of* T.
- *Let* $<$ *be a total strict ordering over* PS*. A* tree-of-BDDs *of a* `CNF` *formula* Σ *given* $<$ *consists of a decomposition tree* T *of* Σ *equipped with a further labelling function* B *such that for every* $n \in N$*,* $B(n)$ *is the* $\mathtt{OBDD}_<$ *representation of* $\exists\overline{Var(n)}.I(\Sigma)$.
We have $Var(T) = \bigcup_{n \in N} Var(n)$ *and* $I(T) = \bigwedge_{n \in N} I(B(n))$*.* `ToB` *denotes the set of all trees-of-BDDs given* $<$.

Clearly, `ToB` is a complete representation language: for every Boolean function there is a `CNF` formula Σ representing it, and thus a tree-of-BDDs T of Σ such that $I(T) = I(\Sigma)$.

The above definition can be simplified and extended, allowing the representation of other formulae than `CNF` ones, and taking advantage of other target languages than $\mathtt{OBDD}_<$ for compiling the labels $B(n)$.

Definition 8 (`ToC`**)** *Let* `C` *be any complete propositional representation language. A* `ToC` *representation is a finite, labelled tree* T*, whose set of nodes is* N*. Each node* $n \in N$ *is labelled with* $Var(n)$*, a subset of* PS *and with a* `C` *representation* $B(n)$.

T *must satisfy:*

- the running intersection property: *for each* $x \in \bigcup_{n \in N} Var(n)$*,* $\{n \in N \mid x \in Var(n)\}$ *forms a connected subtree of* T*, and*
- the global consistency property: *for each* $n \in N$*,* $I(B(n)) \equiv \exists\overline{Var(n)}. \bigwedge_{n \in N} I(B(n))$.

We have $Var(T) = \bigcup_{n \in N} Var(n)$ *and* $I(T) = \bigwedge_{n \in N} I(B(n))$*. The size of a* `ToC` *representation* T *is the size of this tree, plus the sizes of the labels of the nodes of* T *(numbers of variables in* $Var(n)$ *and sizes of* $B(n)$*).*
`ToC` *denotes the set of all* `ToC` *representations.*

Compiling a `CNF` formula Σ into a `ToC` representation T basically consists in computing first a decomposition tree of Σ, then taking advantage of any `CNF`-to-`C` compiler so as to turn the `CNF` $clauses(n)$ formulae (for each node n of the tree) into equivalent `C` representations, and finally to use the well-known message-passing propagation algorithm (see the **Propagate** function in [Subbarayan *et al.*, 2007], which applies also to `ToC` representations) from the leaves of the tree to its root then from the root to the leaves so as to ensure the global consistency property.[2] The running intersection property enables one to replace a global computation on the resulting `ToC` representation T by a number of possibly easier, local computations on the corresponding $B(n)$.

Taking `C` = $\mathtt{OBDD}_<$, we get the $\mathtt{ToOBDD}_<$ language. Within this language, unlike with the $\mathtt{OBDD}_<$ one, a propositional formula may have several equivalent representations. For instance, let $\Sigma = (\neg a \wedge \neg b) \vee (\neg a \wedge c) \vee (b \wedge c)$. Whatever $<$, this formula can be represented by the $\mathtt{ToOBDD}_<$ representation T such that T has a single node n_0, such that $Var(n_0) = Var(\Sigma)$ and $B(n_0)$ is the $\mathtt{OBDD}_<$ formula representing Σ; observing that $\Sigma \equiv (\neg a \vee b) \wedge (\neg b \vee c)$, Σ can also be represented by the $\mathtt{ToOBDD}_<$ representation T such that T has two nodes n_0 and n_1, the root of T is n_0, $Var(n_0) = \{a, b\}$, $Var(n_1) = \{b, c\}$, $B(n_0)$ is the $\mathtt{OBDD}_<$ formula equivalent to $(\neg a \vee b)$, and $B(n_1)$ is the $\mathtt{OBDD}_<$ formula equivalent to $(\neg b \vee c)$. In short, $\mathtt{ToOBDD}_<$ does not offer the property of canonical representation. Clearly, this definition of $\mathtt{ToOBDD}_<$ is close to the previous one `ToB` from [Subbarayan *et al.*, 2007], except that a $\mathtt{ToOBDD}_<$ representation T is defined *per se*, i.e., independently from a given `CNF` formula Σ.

Let us now present some generic properties about `ToC` fragments; such properties are about queries, transformations and succinctness, and are related to similar properties satisfied by the corresponding `C` languages. We first need the following definition:

Definition 9 (TE, CL) *Let* `C` *be any propositional representation language.*

- `C` *satisfies* **TE** *(the* term condition*) iff for every term* γ *over* PS*, a* `C` *representation equivalent to* γ *can be computed in time polynomial in* $|\gamma|$.

[2] This approach can be easily extended to deal with the compilation of any conjunctive representation into a `ToC` representation when compilers to `C` are available.

- C *satisfies* **CL** *(the* clause condition*) iff for every clause* δ *over* PS*, a* C *representation equivalent to* δ *can be computed in time polynomial in* $|\delta|$.

Clearly enough, those conditions are not very demanding and are satisfied by all complete languages considered in [Darwiche and Marquis, 2002], but MODS.

Proposition 1 *Let* C *be any complete propositional representation language.*

1. C *satisfies* **CO** *iff* ToC *satisfies* **CO**.
2. C *satisfies* **VA** *iff* ToC *satisfies* **VA**.
3. C *satisfies* **IM** *iff* ToC *satisfies* **IM**.
4. *If* C *satisfies* **CD**, *then* C *satisfies* **ME** *iff* ToC *satisfies* **ME**.
5. *If* C *satisfies* **CL**, *then* ToC *does not satisfy* **CE** *unless* P = NP.
6. *If* C *satisfies* **CL**, *then* ToC *does not satisfy* **SE** *unless* P = NP.

Points 1. to 4. show that the ToC languages typically satisfy all the queries **CO**, **VA**, **IM** and **ME** (just because the corresponding C languages typically satisfy them and **CD**). Similarly, points 5. and 6. show that the ToC languages typically do not satisfy any of **CE** or **SE** unless P = NP (because the corresponding C languages typically satisfy **CL**). Finally, since every propositional language satisfying **CO** and **CD** also satisfies **CE** (a straightforward extension of Lemma 1.4 from [Darwiche and Marquis, 2002] to any propositional representation language), we get as a corollary to points 1. and 5. that:

Corollary 1 *If* C *satisfies* **CO** *and* **CL**, *then* ToC *does not satisfy* **CD** *unless* P = NP.

Considering other transformations, we obtained the following results which hold for any propositional representation language (hence specifically for the ToC ones):

Proposition 2 *Let* C *be any propositional representation language.*

1. *If* C *satisfies* **CO** *and* **TE** *and* C *does not satisfy* **CE** *unless* P = NP, *then* C *does not satisfy* $\wedge$**BC** *unless* P = NP.
2. *If* C *satisfies* **VA** *and* **TE**, *then* C *does not satisfy* $\vee$**C** *unless* P = NP.
3. *If* C *satisfies* **IM** *and does not satisfy* **CE** *unless* P = NP, *then* C *does not satisfy* $\neg$**C** *unless* P = NP.

These results show that the ToC languages typically satisfy only few transformations among **CD**, $\wedge$**BC**, $\vee$**C** and $\neg$**C** (since the conditions on C listed in Corollary 1 and Proposition 2 are not very demanding).

From the practical side, it is interesting to note that the algorithms **Conditioning**, **Project**, **IsCE**, **IsEQ** reported in [Subbarayan *et al.*, 2007] (Figure 3), for respectively computing the conditioning of a ToOBDD$_<$ representation by a consistent term, computing the projection of a ToOBDD$_<$ representation T on a given set V of variables (or equivalently, forgetting all variables in T except those of V), deciding whether a clause is entailed by a ToOBDD$_<$ representation, deciding whether two ToOBDD$_<$ representations are equivalent, apply to ToC representations as well (the fact that each $B(n)$ of T is an OBDD$_<$ representation is not mandatory for ensuring the correctness of these algorithms).

As to succinctness, we got the following results:

Proposition 3 *Let* C *be any complete propositional representation language.*

1. ToC $\leq_\mathcal{P}$ C.
2. *Let* C' *be any complete propositional fragment. If* C $\leq_s$ C', *then* ToC $\leq_s$ ToC'.
3. *If* C *satisfies* **CL** *and* C' *satisfies* **CE** *then* C' $\not\leq_s^*$ ToC.
4. *If* C *satisfies* **IM**, *then* ToC $\not\leq_s^*$ DNF.

Proposition 3 has many interesting consequences:

- From point 1., we directly get that ToC $\leq_s$ C, and that ToC is complete (since C is). This result *cannot* be strengthened to ToC $<_s$ C in the general case (for every C satisfying $\wedge$**C**, e.g. C = CNF, we can prove that C $\sim_\mathcal{P}$ ToC).
- Point 2. allows one to take advantage of previous results describing how propositional languages C are organized w.r.t. spatial efficiency in order to achieve similar results for the corresponding ToC languages.
- Point 3. implies that the DNNF language, which satisfies **CE**, is typically (i.e., whenever C satisfies **CL**) not more succinct than the corresponding ToC language; hence none of the languages C which are less succinct than DNNF (e.g. C = DNF) can be more succinct than such ToC languages; thus, we get for instance that DNF $\not\leq_s^*$ ToDNF (which together with point 1. shows that ToDNF $<_s^*$ DNF).
- Another consequence of point 3. is that if C satisfies **CL** then DNNF $\not\leq_s^*$ ToC (hence d-DNNF $\not\leq_s^*$ ToC). Together with point 1. this shows ToDNNF to be spatially (strictly) more efficient than DNNF. while keeping **CO** and **ME**.

Finally, an interesting issue is to determine whether, at the instance level, targeting ToC in a compilation process leads always to save space w.r.t. targeting C. The answer is "not always" (even in the cases when, at the language level, we have ToC $<_s^*$ C): the following lemma characterizes a subset of Boolean functions f for which no ToC representation more succinct than its C representation(s) may exist.

Definition 10 (decomposition) *Let* f *be a Boolean function. Let* V_1*, ...,* V_k *be* k *subsets of* PS. $D = \{V_1, \ldots, V_k\}$ *is a* decomposition set *for* f *iff we have* $f = \bigwedge_{i=1}^{k} \exists \overline{V_i}.f$. *A Boolean function* $\bigwedge_{i=1}^{k} f_i$ *(where each* f_i *is a Boolean function) is a* conjunctive decomposition *of* f *iff there exists a decomposition set* $D = \{V_1, \ldots, V_k\}$ *for* f *s.t. for each* $i \in 1 \ldots k$*, we have* $f_i = \exists \overline{V_i}.f$.

Lemma 1 *Let* f *be a Boolean function. Let* δ *be an essential prime implicate of* f*, i.e., a prime implicate of* f *which is not implied by the conjunction of the other prime implicates of* f*. Then for every decomposition set* D *for* f*, there exists* $V \in D$ *such that* $Var(\delta) \subseteq V$.

This lemma shows that when f has an essential prime implicate containing all its variables, no ToC representation of f can be more compact than each of its C representations. This lemma also shows that when f has an essential prime implicate δ such that $\exists\overline{Var(\delta)}.f$ has no C representation of reasonable size, chosing ToC as the target language is not a way to save space.

Finally, Lemma 1 also explains why imposing a fixed decomposition tree T for defining a ToC language is not so a good idea (despite the fact it may offer a property of canonicity of the representations in some cases): either T has a node n such that $Var(n) = \{x_1, \ldots, x_p\}$ (all the variables of interest), and in this case the corresponding ToC language mainly amounts to C, or T does not contain such a node, and in this case the ToC language is incomplete: the Boolean function which is the semantics of the clause $\bigvee_{i=1}^{p} x_i$ cannot be represented in ToC.

4 Back to ToOBDD$_<$ Representations

Let us now fix C to OBDD$_<$ in order to get some further results. Beyond ToOBDD$_<$ we have investigated the properties of U(ToOBDD$_<$) (the union of all ToOBDD$_<$ for each total order $<$ over PS) and of ToOBDD, as target languages for propositional knowledge compilation, along the lines of the KC map. To make the differences between these languages clearer, observe that OBDD representations $B(n), n \in N$ where N is the set of nodes of a given ToOBDD T may rely on different variable orders $<$, while all the OBDD$_<$ representations in a given U(ToOBDD$_<$) are based on the same order. Hence, U(ToOBDD$_<$) is a proper subset of ToOBDD in the general case.

Proposition 4 *The results in Table 1 hold.*

The fact that ToOBDD, U(ToOBDD$_<$), and ToOBDD$_<$ satisfy **CO**, **VA**, **IM**, and **ME** and that none of these languages satisfies any of **CE**, **SE**, **CD**, $\wedge$**BC**, $\wedge$**C**, $\vee$**C** or $\neg$**C**, unless P = NP is a direct corollary of Propositions 1 and 2. Except **CO** and **VA**, all those results concern some issues left open in [Subbarayan *et al.*, 2007]. Especially, unlike [Subbarayan *et al.*, 2007], our (polytime) algorithms for **IM** and **ME** are not based on the message-passing propagation algorithm (the **Propagate** function) which does not run in polynomial time in the general case. Furthermore, contrary to what was expected in [Subbarayan *et al.*, 2007], $\neg$**C** is *not* trivial: the negation of a conjunction of OBDD$_<$ representations is equivalent to the *disjunction* of their negations. We actually showed that the $\neg$**C** transformation on ToOBDD$_<$ cannot be achieved in polynomial time unless P = NP.

As to succinctness, we proved the following results:

Proposition 5

1. *For each* $<$, ToOBDD$_<$ $<_s^*$ OBDD$_<$.
2. *For each* $<$, DNNF $\not\leq_s^*$ ToOBDD$_<$.
3. ToOBDD $\not\leq_s^*$ DNF.
4. ToOBDD $\not\leq_s$ CNF.

Points 1. to 3. are direct consequences of Proposition 3 and results from [Darwiche and Marquis, 2002]). A direct consequence of Proposition 5 is that d-DNNF $\not\leq_s^*$ ToOBDD$_<$. This explains in some sense the space savings which can be offered by ToOBDD$_<$ over d-DNNF and observed empirically as reported in [Subbarayan *et al.*, 2007]. More generally, from Proposition 3 and some results given in [Darwiche and Marquis, 2002] we get that:

	CE	**VA**	**CO**	**IM**	**EQ**	**SE**	**CT**	**ME**
ToOBDD	○	√	√	√	?	○	?	√
U(ToOBDD$_<$)	○	√	√	√	?	○	?	√
ToOBDD$_<$	○	√	√	√	?	○	?	√
OBDD	√	√	√	√	√	√	√	√
OBDD$_<$	√	√	√	√	√	√	√	√

	CD	**FO**	**SFO**	$\wedge$**BC**	$\wedge$**C**	$\vee$**BC**	$\vee$**C**	$\neg$**C**
ToOBDD	○	○*	?	○	●	○	○	○
U(ToOBDD$_<$)	○	○*	?	○	●	○	○	○
ToOBDD$_<$	○	○*	?	○	●	?	○	○
OBDD	√	●	√	○	●	○	●	√
OBDD$_<$	√	●	√	√	●	√	●	√

Table 1: √ means "satisfies", ● means "does not satisfy", ○ means "does not satisfy unless P = NP", and ○* means "does not satisfy unless PH collapses." Results for OBDD$_<$ and OBDD are from [Darwiche and Marquis, 2002] and are given here as a baseline.

Corollary 2 *Unless* PH *collapses,* ToOBDD*,* U(ToOBDD$_<$) *and* ToOBDD$_<$ *are* incomparable w.r.t. succinctness *with the languages* CNF*,* DNF*, and* DNNF.

5 Conclusion

In this paper, the concept of tree-of-BDDs has been extended to any complete propositional representation language C thus leading to the family of ToC languages. A number of generic results are provided, which allow to determine the queries/transformations satisfied by ToC depending on the ones satisfied by C, as well as results about the spatial efficiency of the ToC languages. Focusing on the ToOBDD$_<$ language (and some related languages), we have addressed a number of issues that remained open in [Subbarayan *et al.*, 2007]; especially, we have shown that beyond **CO** and **VA**, ToOBDD$_<$ satisfies **IM** and **ME** but does not satisfy any query among **CE**, **SE**. We have also proved that ToOBDD$_<$ does not satisfy any transformation among **CD**, **FO**, $\wedge$**BC**, $\vee$**C** or $\neg$**C** and that this fragment is not comparable for succinctness w.r.t. any of CNF, DNF and DNNF unless PH collapses.

From this investigation, it turns out that the ToOBDD$_<$ language (and in general the ToC languages) satisfies only few queries and transformations. Subsequently, in applications where some queries/transformations not satisfied by ToOBDD$_<$ must be achieved under some guaranteed response time, considering ToOBDD$_<$ as a target language for KC is not always the best choice. However, imposing further restrictions on C can be a way to recover some of them. Thus, it is easy to show that if C has a linear-time algorithm for **FO**and a linear-time algorithm for $\wedge$**BC**, then ToChas a polytime algorithm for **FO**. If, in addition, C has a polytime algorithm for **CD**, then ToC has a polytime algorithm for **CD**. Furthermore, the fact that many queries/transformations are NP-hard in the general case does not discard ToOBDD$_<$ (and beyond it the ToC languages) as interesting target languages for KC *from the practical side* in the general case.[3] Indeed, if the width of

[3]See [Marquis, 2008] for more details on the distinction between

a $\texttt{ToC}$ representation T, i.e. $max_{n \in N}(\{\mid Var(n) \mid -1\})$, is (upper) bounded by a constant,[4] then provided that the time complexities of forgetting variables in a $\texttt{C}$ representation and conjoining two $\texttt{C}$ representations depend only on the number of variables of those representations (which are quite reasonable assumptions), the time complexity of the **Propagate** function becomes linear in the tree size; as a consequence, many other queries and transformations may become tractable as well; for instance if $\texttt{C}$ satisfies **CD**, we get that both conditioning and clausal entailment can be achieved in polynomial time in the tree size. Thus, from the practical side, as reported in [Subbarayan *et al.*, 2007] (and despite the fact that $\texttt{ToOBDD}_< \not\leq_s \texttt{CNF}$), there are $\texttt{CNF}$ formulae which can be compiled into $\texttt{ToOBDD}_<$ using a reasonable amount of computational resources, while it turned out impossible to generate $\texttt{d-DNNF}$ representations for them. Such empirical results cohere with our succinctness result $\texttt{d-DNNF} \not\leq_s^* \texttt{ToOBDD}_<$. Nevertheless, our result $\texttt{ToOBDD} \not\leq_s^* \texttt{DNNF}$ shows that this empirical evidence can be argued (this result implies that some $\texttt{DNNF}$ representations do not have "small" $\texttt{ToOBDD}_<$ equivalent representations under the standard assumptions of complexity theory), so $\texttt{DNNF}$ remains a very attractive language for the KC purpose.

Our results also suggest a number of $\texttt{ToC}$ languages as quite promising. Consider for instance the $\texttt{ToFBDD}$ language. From our results, it comes easily that $\texttt{ToFBDD}$ satisfies **CO**, **VA**, **IM**, **ME** (hence the same queries as $\texttt{ToOBDD}_<$); since $\texttt{ToFBDD}$ is at least as succinct as $\texttt{ToOBDD}$, it appears as a challenging fragment. Furthermore, a compiler to $\texttt{FBDD}$ is already available (see e.g. www.eecg.utoronto.ca/∼jzhu/fbdduser11.ps). When none of **VA** or **IM** is expected, the $\texttt{ToDNNF}$ language looks also valuable; indeed, from our results we know that $\texttt{ToDNNF}$ satisfies **CO** and **ME**, while being quite compact: $\texttt{ToDNNF} <_s^* \texttt{ToOBDD}_<$ and $\texttt{ToDNNF} <_s^* \texttt{DNNF}$ hold; beyond the spatial dimension, targeting the $\texttt{ToDNNF}$ language may also reduce the on-line computation time needed for achieving queries/transformations based on **Propagate** function (as well as the off-line $\texttt{CNF}$-to-$\texttt{ToC}$ compilation time) since $\texttt{DNNF}$ satisfies **FO**, which is one of the two key operations of the propagation algorithm. The $\texttt{ToDNNF}_T$ language, based on $\texttt{DNNF}_T$ [Pipatsrisawat and Darwiche, 2008], also looks interesting in this respect since it satisfies both **FO** and $\wedge$**BC**, the other key operation of the propagation algorithm.

This is what the "theory" says in some sense about such languages. Going further requires to implement compilers and perform experiments in order to determine whether, from the practical side, representations from those languages can be computed using a reasonable amount of resources. This is an issue for further research. Another perspective for further work is to complete the missing results about queries, transformations and succinctness for the $\texttt{ToC}$ languages and to extend the KC map accordingly. Especially, it would be interesting to characterize some families of propositional formulae each of $\texttt{DNNF}$ and $\texttt{ToOBDD}$ are "effective" on (the proof of Corollary 2 is based on the assumption that PH does not collapse, and is a "non-constructive" proof).

the evaluation of KC languages at the language level and at the practical (or "instance") level.

[4] At the language level, the price to be paid by such a restriction is a lack of expressiveness: none of the languages of $\texttt{ToC}$ representations of width bounded by c (where c is a parameter) is complete (this is a direct consequence of Lemma 1).

References

[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.

[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.

[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.

[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.

[Horiyama and Ibaraki, 2002] T. Horiyama and T. Ibaraki. Ordered binary decision diagrams as knowledge-bases. *Artificial Intelligence*, 136:189–213, 2002.

[Marquis, 2008] P. Marquis. Knowledge compilation: a sightseeing tour. In *Tutorial notes, ECAI'08*, 2008. available on-line.

[Meinel and Theobald, 1998] Ch. Meinel and T. Theobald. *Algorithms and Data Structures in VLSI Design: OBDD - Foundations and Applications*. Springer, 1998.

[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.

[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.

[Tison, 1967] P. Tison. Generalization of consensus theory and application to the minimization of boolean functions. *IEEE Transactions on Electronic Computers*, EC–16:446–456, 1967.

[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.

Appendix

Proof:[Proposition 1]

1. Proposition 3 shows that $\texttt{C} \geq_{\mathcal{P}} \texttt{ToC}$. Hence, it holds that if ToC satisfies **CO**, then C satisfies **CO**. Conversely, a formula Σ is inconsistent iff whatever the set of variables V, we have that $\exists V.\Sigma$ is inconsistent. Accordingly, a ToC representation represents an inconsistent formula Σ iff whatever $n \in N$, the C representation $B(n)$ is inconsistent. The fact that C satisfies **CO** allows us to conclude the proof.

2., 3. From $\texttt{C} \geq_{\mathcal{P}} \texttt{ToC}$ (Proposition 3), we get that if ToC satisfies **VA** (resp. **IM**), then C satisfies **VA** (resp. **IM**). Conversely, let T be a ToC representation of a formula Σ. we have $\Sigma \equiv \Sigma(T) \equiv \bigwedge_{n \in N} B(n)$. So $\Sigma(T)$ is valid iff each C formula $B(n)$ for $n \in N$ is valid. Furthermore, a consistent term γ implies $\Sigma(T)$ iff γ implies each C formula $B(n)$ for $n \in N$. The fact that C satisfies **VA** (resp. **IM**) is enough to conclude the proof.

4. Again, from $\texttt{C} \geq_{\mathcal{P}} \texttt{ToC}$ (Proposition 3), we get that if ToC satisfies **ME**, then C satisfies **ME**. Conversely, we can design an output-polynomial time algorithm for computing the models of a ToC representation T over $Var(T)$. This algorithm is a recursive algorithm taking as inputs a ToC representation T and a term γ such as $Var(\gamma) \subseteq Vars(n_0)$ (where n_0 is the root of T). Initially, γ is the empty term and the current node n is n_0. $B(n)$ is conditioned by γ (in polynomial time, since C satisfies **CD**). The algorithm then computes the models ω of the conditioning $B(n)|\gamma$ of $B(n)$ by γ (in output-polynomial time, since C satisfies **ME**). If n is a leaf node, then the algorithm returns this set of models. Otherwise, for each model ω $B(n)|\gamma$, the procedure is recursively called on each of the children of n (which is itself the root of ToC representations) with input $\gamma = \omega$. This returns as many sets of models Ω_i as the number of children of n. Thanks to the running intersection property, the cartesian product P_ω of these Ω_i together with $\{\omega\}$ is the set of models of $\Sigma(T_n)$ over $Var(T_n)$, where T_n is the subtree of T rooted at n: once conditioned, the children of n correspond to representations on disjoint sets of variables, so that the models of $\Sigma(T_n)$ can be generated in a backtrack-free fashion. The algorithm finally returns the union, over the ω, of the P_ω.

5. By reduction from CNF-SAT. Let $\alpha = \bigwedge_{i=1}^{k} \delta_i$ be a CNF formula such that $Var(\alpha) = \{x_1, \ldots, x_p\}$. Let $\alpha' = \bigwedge_{i=1}^{k} \neg holds_i \vee \delta_i$ be a CNF formula such that each $holds_i$ ($i \in 1 \ldots k$) is a fresh variable from PS, not occurring in $Var(\alpha)$. Each $holds_i$ ($i \in 1 \ldots k$) can be viewed as the name given to the clause δ_i. Since ToC satisfies **CL**, we can associate to α in polynomial time the following ToC representation T of α': the set of nodes of T is $N = \{n_0, n_1, \ldots, n_k\}$, n_0 is the root of T and the k remaining nodes are the k children of n_0; furthermore, we have $Var(n_0) = \{x_1, \ldots, x_p\}$ and for every $i \in 1 \ldots k$, $Var(n_i) = \{holds_i\} \cup Var(\delta_i)$. Now, by construction α' is consistent and every clause which is a logical consequence of α' contains one of the $\neg holds_i$ as a literal. Hence $\exists\{holds_i, i = 1, k\}\alpha'$ is a valid formula, and whatever j, the formula $\exists\{holds_i, i = 1, k\} \setminus \{j\}\alpha'$ is equivalent to the clause $\neg holds_j \vee \delta_j$. Obviously, we can compute in time polynomial in the size of α the remaining labels $B(n_0) = 1$ and $B(n_i)$ ($i \in 1 \ldots k$) as the C representation of $\neg holds_i \vee \delta_i$ when C satisfies **CL**. We also associate to α in polynomial time the clause $\delta = \bigvee_{i=1}^{k} \neg holds_i$. Finally, α is consistent iff $\alpha' \not\models \delta$ iff $\Sigma(T) \not\models \delta$. This completes the proof.

6. Comes easily from the fact that ToC does not satisfy **CE** unless P = NP, plus the fact that a ToC representation of any clause $\delta = \bigvee_{i=1}^{j} l_i$ can be generated in polynomial time from the size of δ when ToC satisfies **CL**.

■

Proof:[Proposition 2]

1. $\wedge$**BC**. Towards a contradiction. Assume that C satisfies $\wedge$**BC**. Then since it satisfies **TE**, one can compute in polynomial time a C representation of $T \wedge \gamma$ from a C representation T and a term γ, and one can decide in polynomial time whether it is consistent since C satisfies **CO**. But a non-valid clause δ is an implicate of T iff $T \wedge \gamma$ is inconsistent where the term γ can be trivially computed in linear time from $\neg\delta$ and is equivalent to it. This completes the proof.

   By reduction from CNF-SAT. Consider again the reduction given in point 5. of Proposition 1. We have that α is consistent iff $\alpha' \not\models \delta$ iff $\Sigma(T) \not\models \delta$. This is also equivalent to $\Sigma(T) \wedge \neg\delta$ is consistent. The fact that C satisfies **CO** and **TE** completes the proof.

2. $\vee$**C**. When C satisfies **TE**, every term γ has a C representation of size polynomial in the size of γ Since DNF does not satisfy **VA** unless P = NP, if C satisfies it, it cannot be the case that C satisfies $\vee$**C** unless P = NP.

3. $\neg$**C**. Towards a contradiction, assume that C satisfies $\neg$**C**; then from any C representation T of a formula Σ we can compute in polynomial time a C representation T' of $\neg\Sigma$. Now, for every formula Σ and every clause δ, we have that $\Sigma \models \delta$ iff the term γ which can be trivially computed in linear time from $\neg\delta$ and is equivalent to it is an implicant of $\neg\Sigma$, hence of T'. The fact that C satisfies **IM** concludes the proof.

■

Proof:[Proposition 3]

1. Every C formula Σ can associated in linear time to its ToC representation T such that T has a single node n_0, $Var(n_0) = Var(\Sigma)$ and $B(n_0)$ is the C formula equivalent to Σ.

2. To every ToC′ representation T' of a formula Σ, we can associate via a polysize function a ToC representation T of Σ. The tree T coincides with T', except that each node n of this tree is now labelled by a C formula $B(n)$

equivalent to the C' formula $B'(n)$ labelling n in T'; indeed, since C leq_s C' , there exists a polysize function f such that each $B(n)$ can be computed via f from $B'(n)$.

3. The proof is close to the one used to show that ToC does not satisfy **CE**, unless P = NP when C satisfies **CL** (point 5. of Proposition 1). Let p be any non-negative integer. Let Σ_p^{max} be the CNF formula

$$\bigwedge_{\delta_i \in 3-C_p} \neg holds_i \vee \delta_i$$

where $3 - C_n$ is the set of all 3-literal clauses that can be generated from $\{x_1, \ldots, x_p\}$ and the $holds_i$ are new variables, not among $x_1, \ldots, x_p$. The size $|\Sigma_p^{max}|$ of Σ_p^{max} is in $\mathcal{O}(p^3)$ since Σ_p^{max} contains $\mathcal{O}(p^3)$ 4-literal clauses. Especially, the number k of clauses of Σ_p^{max} is cubic in p. Now, we associate to Σ_p^{max} in polynomial time the following ToC representation T of it: the set of nodes of T is $N = \{n_0, n_1, \ldots, n_k\}$, n_0 is the root of T and the k remaining nodes are the k children of n_0; furthermore, we have $Var(n_0) = \{x_1, \ldots, x_p\}$ and for every $i \in 1 \ldots k$, $Var(n_i) = \{holds_i\} \cup Var(\delta_i)$. Now, by construction, every clause which is a logical consequence of Σ_p^{max} contains one of the $\neg holds_i$ as a literal. Hence forgetting every $holds_i$ in Σ_p^{max} leads to a valid formula, and forgetting all the $holds_i$ in Σ_p^{max} except one, say $holds_j$, leads to a formula equivalent to the clause $\neg holds_j \vee \delta_j$. We can compute in time polynomial in the size of Σ_p^{max} the remaining labels $B(n_0) = 1$ and $B(n_i)$ ($i \in 1k$) as the C representation of the clause $\neg holds_i \vee \delta_i$, since C satisfies **CL**.
Each 3-CNF formula α_p built up from the set of variables $\{x_1, \ldots, x_p\}$ is in bijection with the subset S_{α_p} of the variables $holds_i$ s.t. δ_i is a clause of α_p if and only if $holds_i \in S_{\alpha_p}$ (each variable $holds_i$ can be viewed as the name of the clause where it appears, hence selecting a clause just amounts to select its name). Let δ_{α_p} be the clause

$$\bigvee_{holds_i \in S_{\alpha_p}} \neg holds_i.$$

It is easy to check that α_p is inconsistent iff

$$\Sigma_p^{max} \models \delta_{\alpha_p}.$$

Suppose now that C' is at least at succinct as ToC. Then using a polysize compilation function $comp$ we could associate to T an equivalent C' formula $comp(T)$. Since C' satisfies **CE**, for every clause δ, determining whether $comp(T) \models \delta$ can be done in (deterministic) polynomial time. Then we would be able to determine whether any 3-CNF formula α is satisfiable using a deterministic Turing machine with a polynomial advice A: if $|Var(\alpha)| = p$, then the machine loads

$$A(n) = comp(T).$$

Once this is done, it determines whether δ_α is entailed by $comp(T)$, which is in P. Since 3-SAT is complete for NP, this would imply NP $\subseteq$ P/poly, and, as a consequence, the polynomial hierarchy would collapse at the second level.

4. This result is in some sense dual to point 3. Let p be any non-negative integer. Let Σ_p^{max} be the CNF formula

$$\bigwedge_{\delta_i \in 3-C_p} \neg holds_i \vee \delta_i$$

where $3 - C_n$ is the set of all 3-literal clauses that can be generated from $\{x_1, \ldots, x_p\}$ and the $holds_i$ are new variables, not among $x_1, \ldots, x_p$. The size $|\Sigma_p^{max}|$ of Σ_p^{max} is in $\mathcal{O}(p^3)$ since Σ_p^{max} contains $\mathcal{O}(p^3)$ 4-literal clauses. Especially, the number k of clauses of Σ_p^{max} is cubic in p. Obviously enough, each 3-CNF formula α_p built up from the set of variables $\{x_1, \ldots, x_p\}$ is in bijection with the subset S_{α_p} of the variables $holds_i$ s.t. δ_i is a clause of α_p if and only if $holds_i \in S_{\alpha_p}$ (each variable $holds_i$ can be viewed as the name of the clause where it appears, hence selecting a clause just amounts to selecting its name). Let δ_{α_p} be the clause

$$\bigvee_{holds_i \in S_{\alpha_p}} \neg holds_i.$$

It is easy to check that α_p is inconsistent if and only if $\Sigma_p^{max} \models \delta_{\alpha_p}$ if and only if $\neg\delta_{\alpha_p} \models \neg\Sigma_p^{max}$.
Suppose now that ToC is at least at succinct as DNF. Then using a polysize compilation function $comp$ we could associate to the DNF formula $\neg\Sigma_p^{max}$ an equivalent ToC representation T. If C satisfies **IM**, then from point 3. of Proposition 1, ToC satisfies **IM** as well. So, for every term γ, determining whether γ is an implicant of the formula represented by T can be done in (deterministic) polynomial time. Then we would be able to determine whether any 3-CNF formula α is satisfiable using a deterministic Turing machine with a polynomial advice A: if $|Var(\alpha)| = p$, then the machine loads

$$A(n) = T.$$

Once this is done, it determines whether $\gamma_\alpha = \neg\delta_\alpha$ is an implicant of T, which is in P. Since 3-SAT is complete for NP, this would imply NP $\subseteq$ P/poly, and, as a consequence, the polynomial hierarchy would collapse at the second level.

■

Proof:[Lemma 1]

Towards a contradiction, assume that there exists a decomposition set $D = \{V_1, \ldots, V_k\}$ for Σ such that for each $i \in 1 \ldots k$, $Var(\delta) \not\subseteq V_i$. Since for each $i \in 1 \ldots k$,

$$PI(\exists\overline{V_i}.\Sigma) = \{\delta' \in PI(\Sigma) \mid Var(\delta') \subseteq V_i\},$$

we have that

$$\delta \notin \bigcup_{i=1}^{k} PI(\exists\overline{V_i}.\Sigma).$$

Since $\bigcup_{i=1}^{k} PI(\exists\overline{V_i}.\Sigma)$ is a subset of $PI(\Sigma)$ not containing δ and since the conjunction of all prime implicates of Σ except δ does not imply δ, by monotony of $\models$, we get that $\bigwedge_{i=1}^{k} PI(\exists\overline{V_i}.\Sigma) \not\models \delta$. However,

$$\bigwedge_{i=1}^{k} PI(\exists\overline{V_i}.\Sigma) \equiv \Sigma$$

whenever $\{V_1, \ldots, V_k\}$ is a decomposition set for Σ, so we must also have

$$\bigwedge_{i=1}^{k} PI(\exists \overline{V_i}.\Sigma) \models \delta$$

since δ is an implicate of Σ, contradiction. ■

Proof: [Proposition 4]

- **Queries.**
  - As to **CO**, **VA**, **IM**, and **ME**, given the inclusions $\texttt{ToOBDD}_< \subseteq \texttt{U}(\texttt{ToOBDD}_<) \subseteq \texttt{ToOBDD}$, it is enough to consider the case of ToOBDD representations. Since OBDD satisfies each of **CO**, **VA**, **IM**, **ME**, and **CD**, points 1. to 4. of Proposition 1 are enough to get the expected conclusion.
  - As to **CE**, and **SE**, given the inclusions $\texttt{ToOBDD}_< \subseteq \texttt{U}(\texttt{ToOBDD}_<) \subseteq \texttt{ToOBDD}$, it is enough to consider the case of $\texttt{ToOBDD}_<$ representations. Since $\texttt{ToOBDD}_<$ satisfies **CL**, points 5. and 6. of Proposition 1 concludes the proof.
- **Transformations.**
  - **CD**. Given the inclusions $\texttt{ToOBDD}_< \subseteq \texttt{U}(\texttt{ToOBDD}_<) \subseteq \texttt{ToOBDD}$, it is enough to consider the case of $\texttt{ToOBDD}_<$ representations. Since $\texttt{OBDD}_<$ satisfies **CO** and **CL**, Corollary 1 gives the result.
  - **FO**. None of $\texttt{ToOBDD}_<$ (whatever $<$), ToOBDD or $\texttt{U}(\texttt{ToOBDD}_<)$ satisfies **FO** unless PH collapses. The proof is similar to the one used to show that $\texttt{OBDD}_<$ does not satisfy **FO** [Darwiche and Marquis, 2002]. To any DNF formula $\Sigma = \bigvee_{i=1}^{k} \gamma_i$, we associate in polynomial time the $\texttt{ToOBDD}_<$ representation T such that the set of nodes of T is $\{n_0\}$ and $B(n_0)$ is the $\texttt{OBDD}_<$ formula equivalent to Δ^1 inductively defined by:
    - $\Delta^k = OBDD_<(\gamma_k)$, i.e. the $\texttt{OBDD}_<$ formula equivalent to the term γ_k,
    - Δ^i is the $\texttt{OBDD}_<$ of the form $(new_i \wedge OBDD_<(\gamma_i)) \vee (\neg new_i \wedge \Delta^{i+1})$ for each $i \in 1 \ldots k-1$.

    Here each new_i ($i \in 1 \ldots k$) is a fresh variable not occurring in Σ. We assume that the new_i variables ($i \in 1 \ldots k$) are earlier than all other variables w.r.t. $<$. We have $\Sigma \equiv \exists\{new_1, \ldots, new_k\}.\Sigma(T)$. Obviously, T also is a ToOBDD representation of Σ and a $\texttt{U}(\texttt{ToOBDD}_<)$ representation of Σ. If $\texttt{ToOBDD}_<$ (resp. ToOBDD, $\texttt{U}(\texttt{ToOBDD}_<)$) would satisfy **FO**, then we could turn in polynomial time the DNF formula Σ into an equivalent $\texttt{ToOBDD}_<$ (resp. ToOBDD, $\texttt{U}(\texttt{ToOBDD}_<)$) representation T. Given the inclusions $\texttt{ToOBDD}_< \subseteq \texttt{U}(\texttt{ToOBDD}_<) \subseteq \texttt{ToOBDD}$, since $\texttt{U}(\texttt{ToOBDD}_<) \not\leq_s$ DNF unless PH collapses, this would make PH to collapse.
  - $\wedge$**BC**. Each of $\texttt{ToOBDD}_<$ (whatever $<$), ToOBDD and $\texttt{U}(\texttt{ToOBDD}_<)$ satisfies **CO** (see above) and **TE** (since this is the case for $\texttt{OBDD}_<$), but does not satisfy **CE** unless P = NP (see above). Then point 1. of Proposition 2 achieves the proof.
  - $\wedge$**C**. Each of $\texttt{ToOBDD}_<$ (whatever $<$), ToOBDD and $\texttt{U}(\texttt{ToOBDD}_<)$ satisfies **CL**. If any of them would satisfy $\wedge$**C**, then CNF would be polynomially translatable into it. This would make this language at least at succinct as CNF, and this would contradict item 4. of Proposition 5 given the inclusions $\texttt{ToOBDD}_< \subseteq \texttt{U}(\texttt{ToOBDD}_<) \subseteq \texttt{ToOBDD}$.
  - $\vee$**BC**. It is known that checking whether the conjunction of two $\texttt{OBDD}_<$ formulae α and β (w.r.t. two different variable orderings $<$) is consistent is NP-complete (see Lemma 8.14 in [Meinel and Theobald, 1998]). Given that each $\texttt{OBDD}_<$ language (whatever $<$) satisfies $\neg$**C**, unless P = NP, there is no polynomial-time algorithm for deciding whether the disjunction of two $\texttt{OBDD}_<$ formulae (w.r.t. two different variable orderings $<$) is valid. Now, every $\texttt{OBDD}_<$ language (whatever $<$) is polynomially translatable into ToOBDD and into $\texttt{U}(\texttt{ToOBDD}_<)$, given that $\texttt{ToOBDD} \subseteq \texttt{U}(\texttt{ToOBDD}_<)$. Since each of ToOBDD and $\texttt{U}(\texttt{ToOBDD}_<)$ satisfies **VA**, none of them can satisfy $\vee$**BC** unless P = NP.
  - $\vee$**C**. Each of $\texttt{ToOBDD}_<$ (whatever $<$), ToOBDD and $\texttt{U}(\texttt{ToOBDD}_<)$ satisfies **VA** (see above) and **TE** (since this is the case for $\texttt{OBDD}_<$). Then point 2. of Proposition 2 achieves the proof.
  - $\neg$**C**. Each of $\texttt{ToOBDD}_<$ (whatever $<$), ToOBDD and $\texttt{U}(\texttt{ToOBDD}_<)$ satisfies **IM** (see above) but does not satisfy **CE** unless P = NP (see above). Then point 3. of Proposition 2 achieves the proof.

■

Proof: [Proposition 5]

Points 1. to 3. are direct consequences of Proposition 3.

4. Consider the formula

$$\Sigma = (\bigwedge_{i=1}^{m}(\bigvee_{j=1}^{m} \neg x_{i,j})) \wedge (\bigwedge_{j=1}^{m}(\bigvee_{i=1}^{m} \neg x_{i,j})) \wedge (\bigvee_{i=1}^{m}\bigvee_{j=1}^{m} x_{i,j})$$

over m^2 variables with $m > 1$. Every resolvent from a pair of clauses from Σ is a valid clause and no clause from Σ is implied by another clause from Σ. Hence, the correctness of any resolution-based algorithm for computing prime implicates (like Tison's one [Tison, 1967]) ensures that Σ is a PI formula (hence a CNF formula). Since

$$(\bigwedge_{i=1}^{m}(\bigvee_{j=1}^{m} \neg x_{i,j})) \wedge (\bigwedge_{j=1}^{m}(\bigvee_{i=1}^{m} \neg x_{i,j}))$$

is consistent and negative (i.e., it contains only negative literals), no positive clause like $\bigvee_{i=1}^{m}\bigvee_{j=1}^{m} x_{i,j}$ can be a logical consequence of it. Hence $\bigvee_{i=1}^{m}\bigvee_{j=1}^{m} x_{i,j}$ is an essential prime implicate of Σ. From Lemma 1, we get that for every decomposition set D for Σ, there exists $V \in D$ such that $Var(\bigvee_{i=1}^{m}\bigvee_{j=1}^{m} x_{i,j}) \subseteq V$. This shows that in every $\texttt{ToOBDD}_<$ representation T of Σ there is a node n such that $Var(n) = Var(\Sigma)$. What remains to be shown is that the size of every OBDD formula

$$B(n) \equiv \exists \overline{Var(\Sigma)}.\Sigma \equiv \Sigma$$

is not polynomial in the size of Σ. To this end, we consider the proof of Lemma 3.5 from [Horiyama and Ibaraki, 2002], showing that every OBDD formula equivalent to

$$(\bigwedge_{i=1}^{m}(\bigvee_{j=1}^{m} \neg x_{i,j})) \wedge (\bigwedge_{j=1}^{m}(\bigvee_{i=1}^{m} \neg x_{i,j}))$$

has a size in $\Omega(2^{\frac{m}{\sqrt{2}}})$ (the proof of [Horiyama and Ibaraki, 2002]still applies to our formula Σ since the proposed fooling set A containing $2^{\frac{m}{\sqrt{2}}}$ assignments also is a fooling set when Σ is considered instead of $(\bigwedge_{i=1}^{m}(\bigvee_{j=1}^{m} \neg x_{i,j})) \wedge (\bigwedge_{j=1}^{m}(\bigvee_{i=1}^{m} \neg x_{i,j}))$).

■