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 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 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, OBDD\_<, OBDD (the union of all 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 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 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 ToOBDD\_< does not satisfy any transformation among CD, FO, \_\_BC, \_\_VC or \_\_C. Among other succinctness results, we prove that the 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 formulæ. 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 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 sup-
port $Ω(f)$ of $f$ is the set of all assignments $ω$ of $Var(f)$ to
Boolean values such that $f(ω) = 1$. For any $X \subseteq PS$, we
note by $X$ the set $PS \setminus X$. The set of Boolean functions is
equipped with the three standard internal laws, $\land$, $\lor$ and $\lnot$.
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 $ω_{Var(f)} \setminus X$
of $Var(f) \setminus X$ iff there exists an assignment $ω_X$ of $X$ such
that $f(ω_{Var(f)} \setminus X, ω_X) = 1$.

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

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

Clearly enough, formulae (viewed as words) from a stan-
dard propositional language are representations of Boolean
functions. The size of such a formula is the number of sym-
bols 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 $DAG-NNF$ language [Darwiche and Marquis, 2002]
is also a complete graph-based representation language of
Boolean functions. Distinguished formulae from $DAG-NNF$
are the literals over $PS$, the clauses (a clause is a finite dis-
junction of literals or the Boolean constant $\top$) and the terms
(a term is a finite conjunction of literals or the Boolean con-
stant $\bot$). We assume the reader to be familiar with the
$DAG-NNF$ fragments $DNNF$, d–$DNNF$, CNF, DNF, FBDD, OBDD, OBDDc, OBDD, MODS, etc.

Obviously, all the logical notions pertaining to formulae
viewed up to logical equivalence can be easily extended to
any representation language $C$ of Boolean functions. For in-
stance, an assignment $ω$ of $Var(α)$ to Boolean values is said
to be a model of a $C$ representation $α$ over $Var(α)$ iff
$I(α)(ω) = 1$. Similarly, two representations $α$ and $β$ (pos-
sibly from different representation formalisms) are said to
be equivalent, noted $α \equiv β$, when they represent the same

Boolean function. A $C$ 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 conse-
quence of $β$, noted $β \models α$, iff $Ω(I(β)) \subseteq Ω(I(α))$.

We are now ready to extend the notions of queries, trans-
formations 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 $C$ denote a propositional rep-
resentation language.

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

Definition 4 (transformations) Let $C$ denote a proposi-
tional representation language.

- $C$ satisfies $CD$ iff there exists a polytime algorithm that
  maps every $C$ representation $α$ and every consistent term
  $γ$ to a $C$ representation $β$ of the restriction of $I(α)$ to
  $I(γ)$, i.e., $Var(β) = Var(α) \setminus Var(γ)$ and $I(β) =
  \exists Var(γ). (I(α) \land I(γ))$.
- $C$ satisfies $FO$ iff there exists a polytime algorithm that
  maps every $C$ representation $α$ and every subset $X$ of
  $V ar$ variables from $PS$ to a $C$ representation of $\exists X.I(α)$. If
  the property holds for each singleton $X$, we say that $C$
  satisfies $SFO$.
- $C$ satisfies $\land C$ (resp. $\lor C$) iff there exists a polytime
  algorithm that maps every finite set of $C$ representations
  $α_1, …, α_n$ to a $C$ representation of $I(α_1) \land … \land I(α_n)$
  (resp. $I(α_1) \lor … \lor I(α_n)$).
- $C$ satisfies $\land BC$ (resp. $\lor BC$) iff there exists a polytime
  algorithm that maps every pair of $C$ representations $α$
  and $β$ to a $C$ representation of $I(α) \land I(β)$ (resp. $I(α) \lor
  I(β)$).
- $C$ satisfies $\neg C$ iff there exists a polytime algorithm that
  maps every $C$ representation $α$ to a $C$ representation of
  $\neg I(α)$. 
Definition 5 (succinctness) Let $C_1$ and $C_2$ be two representation languages. $C_1$ is at least as succinct as $C_2$, noted $C_1 \leq_s C_2$, iff there exists a polynomial $p$ such that for every $C_2$ representation $\alpha$ there exists an equivalent $C_1$ representation $\beta$ where $|\beta| \leq p(|\alpha|)$.

$\sim_s$ is the symmetric part of $\leq_s$ defined by $C_1 \sim_s C_2$ iff $C_1 \leq_s C_2$ and $C_2 \leq_s C_1$. $\prec_s$ is the asymmetric part of $\leq_s$ defined by $C_1 \prec_s C_2$ iff $C_1 \leq_s C_2$ and $C_2 \not\leq_s C_1$. Finally, $C_1 \leq_s^\ast C_2$ (resp. $C_1 \prec_s^\ast C_2$) means that $C_1 \leq_s C_2$ (resp. $C_1 \prec_s C_2$) 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 $C_1$ and $C_2$ be two representation languages. $C_1$ is polynomially translatable into $C_2$, noted $C_1 \geq_p C_2$, iff there exists a polytime algorithm $A$ such that for every $C_1$ representation $\alpha$ $A(\alpha)$ is a $C_2$ representation such that $A(\alpha) \equiv_\alpha$.

Like $\geq_s \geq_p$ 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 $C_1$ and $C_2$, if $C_1 \geq_p C_2$, then $C_1 \geq_s C_2$ (but the converse does not hold in general). We note by $\sim_p$ the symmetric part of $\geq_p$.

3 The $\text{Toc}$ 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 $\text{CNF}$ formula $\Sigma$ is a (finite) labelled tree $T$ whose set of nodes is $N$. Each node $n \in N$ is labelled with $\text{Var}(n)$, a subset of $\text{Var}(\Sigma)$. For each $n \in N$, let $\text{clauses}(n) = \{\text{clause } \delta \text{ of } \Sigma \mid \text{Var}(\delta) \subseteq \text{Var}(n)\}$. $T$ satisfies two conditions: for every clause $\delta$ of $\Sigma$ there exists $n \in N$ such that $\delta \in \text{clauses}(n)$, and for every $x \in \text{Var}(\Sigma)$, $\{n \in N \mid x \in \text{Var}(n)\}$ forms a connected subtree of $T$.

- Let $\prec$ be a total strict ordering over $PS$. A tree-of-BDDs of a $\text{CNF}$ formula $\Sigma$ given $\prec$ consists of a decomposition tree $T$ of $\Sigma$ equipped with a further labelling function $B$ such that for every $n \in N$, $B(n)$ is the $\text{OBDD} < \prec$ representation of $\exists \text{Var}(n).I(\Sigma)$. We have $\text{Var}(T) = \bigcup_{n \in N} \text{Var}(n)$ and $I(T) = \bigwedge_{n \in N} I(B(n))$. $\text{Toc}$ denotes the set of all trees-of-BDDs given $\prec$.

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

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

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

$T$ must satisfy:

- the running intersection property: for each $x \in \bigcup_{n \in N} \text{Var}(n)$, $\{n \in N \mid x \in \text{Var}(n)\}$ forms a connected subtree of $T$, and

- the global consistency property: for each $n \in N$, $I(B(n)) \equiv \exists \text{Var}(n), \bigwedge_{n \in N} I(B(n))$.

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

Compiling a $\text{CNF}$ formula $\Sigma$ into a $\text{Toc}$ representation $T$ basically consists in computing first a decomposition tree of $\Sigma$, then taking advantage of any $\text{CNF}$-to-$C$ compiler so as to turn the $\text{CNF}$ 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 $\text{Toc}$ 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. The running intersection property enables one to replace a global computation on the resulting $\text{Toc}$ representation $T$ by a number of possibly easier local computations on the corresponding $B(n)$.

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

Let us now present some generic properties about $\text{Toc}$ 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 $\gamma$ over $PS$, a $C$ representation equivalent to $\gamma$ can be computed in time polynomial in $|\gamma|$.

---

2This approach can be easily extended to deal with the compilation of any conjunctive representation into a $\text{Toc}$ representation when compilers to $C$ are available.
• $C$ satisfies $\text{CL}$ (the clause condition) iff for every clause $\delta$ over $PS$, a $C$ representation equivalent to $\delta$ 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 $\text{MODS}$.

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

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

Points 1. to 4. show that the $\text{ToC}$ languages typically satisfy all the queries $\text{CO}$, $\text{VA}$, $\text{IM}$ and $\text{ME}$ (just because the corresponding $C$ languages typically satisfy them and $\text{CD}$). Similarly, points 5. and 6. show that the $\text{ToC}$ languages typically do not satisfy any of $\text{CE}$ or $\text{SE}$ unless $P = \text{NP}$ (because the corresponding $C$ languages typically satisfy $\text{CL}$).

Finally, since every propositional language satisfying $\text{CO}$ and $\text{CD}$ also satisfies $\text{CE}$ (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 $\text{CO}$ and $\text{CL}$, then $\text{ToC}$ does not satisfy $\text{CD}$ unless $P = \text{NP}$.

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

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

1. If $C$ satisfies $\text{CO}$ and $\text{TE}$ and $C$ does not satisfy $\text{CE}$ unless $P = \text{NP}$, then $C$ does not satisfy $\land BC$ unless $P = \text{NP}$.
2. If $C$ satisfies $\text{VA}$ and $\text{TE}$, then $C$ does not satisfy $\lor C$ unless $P = \text{NP}$.
3. If $C$ satisfies $\text{IM}$ and does not satisfy $\text{CE}$ unless $P = \text{NP}$, then $C$ does not satisfy $\lnot C$ unless $P = \text{NP}$.

These results show that the $\text{ToC}$ languages typically satisfy only few transformations among $\text{CD}$, $\land BC$, $\lor C$ and $\lnot C$ (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 $\text{Conditioning}$, $\text{Project}$, $\text{IsCE}$, $\text{IsEQ}$ reported in [Subbarayan et al., 2007] (Figure 3), for respectively computing the conditioning of a $\text{ToOBDD}_<$ representation by a consistent term, computing the projection of a $\text{ToOBDD}_<$ 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 $\text{ToOBDD}_<$ representation, deciding whether two $\text{ToOBDD}_<$ representations are equivalent, apply to $\text{ToC}$ representations as well (the fact that each $B(n)$ of $T$ is an $\text{OBDD}_<$ 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. $\text{ToC} \leq_P C$.
2. Let $C'$ be any complete propositional fragment. If $C \leq_{s} C'$, then $\text{ToC} \leq_{s} \text{ToC'}$.
3. If $C$ satisfies $\text{CL}$ and $C'$ satisfies $\text{CE}$ then $C' \not<_{s} \text{ToC}$.
4. If $C$ satisfies $\text{IM}$, then $\text{ToC} \not<_{s} \text{DNF}$.

Proposition 3 has many interesting consequences:

• From point 1., we directly get that $\text{ToC} \leq_{s} C$, and that $\text{ToC}$ is complete (since $C$ is). This result cannot be strengthened to $\text{ToC} \leq_{s} C$ in the general case (for every $C$ satisfying $\land C$, e.g. $C = \text{CNF}$, we can prove that $C \not<_{p} \text{ToC}$).

• 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 $\text{ToC}$ languages.

• Point 3. implies that the $\text{DNNF}$ language, which satisfies $\text{CE}$, is typically (i.e., whenever $C$ satisfies $\text{CL}$) not more succinct than the corresponding $\text{ToC}$ language; hence none of the languages $C$ which are less succinct than $\text{DNF}$ (e.g. $C = \text{DNF}$) can be more succinct than such $\text{ToC}$ languages; thus, we get for instance that $\text{DNF} \not<_{s} \text{ToDNNF}$ (which together with point 1. shows that $\text{ToDNNF} \not<_{s} \text{DNF}$).

• Another consequence of point 3. is that if $C$ satisfies $\text{CL}$ then $\text{DNNF} \not<_{s} \text{ToC}$ (hence $\text{DNNF} \not<_{s} \text{ToC}$). Together with point 1. this shows $\text{ToDNNF}$ to be spatially (strictly) more efficient than $\text{DNF}$, while keeping $\text{CO}$ and $\text{ME}$.

Finally, an interesting issue is to determine whether, at the instance level, targeting $\text{ToC}$ 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 $\text{ToC} \leq_{s} C$); the following lemma characterizes a subset of Boolean functions $f$ for which no $\text{ToC}$ representation more succinct than its $C$ representation(s) may exist.

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

Lemma 1 Let $f$ be a Boolean function. Let $\delta$ 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 $\text{Var}(\delta) \subseteq V$. 


This lemma shows that when \( f \) has an essential prime implicate containing all its variables, no \( \text{ToC} \) representation of \( f \) can be more compact than each of its \( \text{C} \) representations. This lemma also shows that when \( f \) has an essential prime implicate \( \delta \) such that \( \exists \text{Var}(\delta). f \) has no \( \text{C} \) representation of reasonable size, choosing \( \text{ToC} \) 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 \( \text{ToC} \) 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 \( \text{Var}(n) = \{ x_1, \ldots, x_p \} \) (all the variables of interest), and in this case the corresponding \( \text{ToC} \) language mainly amounts to \( \text{C} \), or \( T \) does not contain such a node, and in this case the \( \text{ToC} \) language is incomplete: the Boolean function which is the semantics of the clause \( \bigvee_{i=1}^p x_i \) cannot be represented in \( \text{ToC} \).

4 Back to \( \text{ToOBDD}_< \) Representations

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

Proposition 4 The results in Table 1 hold.

The fact that \( \text{ToOBDD}_<, U(\text{ToOBDD}_<), \) and \( \text{ToOBDD}_< \) satisfy \( \text{CO}, \text{VA}, \text{IM}, \) and \( \text{ME} \) and that none of these languages satisfies any of \( \text{CE}, \text{SE}, \text{CD}, \land \text{BC}, \land \text{C}, \lor \text{C} \) or \( \lnot \text{C} \), unless \( \text{P} = \text{NP} \) is a direct corollary of Propositions 1 and 2. Except \( \text{CO} \) and \( \text{VA} \), all those results concern some issues left open in [Subbarayan et al., 2007]. Especially, unlike [Subbarayan et al., 2007], our (polynomial) algorithms for \( \text{IM} \) and \( \text{ME} \) 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], \( \lnot \text{C} \) is not trivial: the negation of a conjunction of \( \text{OBDD}_< \) representations is equivalent to the disjunction of their negations. We actually showed that the \( \lnot \text{C} \) transformation on \( \text{ToOBDD}_< \) cannot be achieved in polynomial time unless \( \text{P} = \text{NP} \).

As to succinctness, we proved the following results:

Proposition 5

1. For each \(<\), \( \text{ToOBDD}_< \) \( \not\prec \text{OBDD}_< \).
2. For each \(<\), \( \text{DNNF} \not\prec \text{ToOBDD}_< \).
3. \( \text{ToOBDD} \not\prec \text{DNNF} \).
4. \( \text{ToOBDD} \not\prec \text{CNF} \).

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 \( \text{d}-\text{DNNF} \not\prec \text{ToOBDD}_< \). This explains in some sense the space savings which can be offered by \( \text{ToOBDD}_< \) over \( \text{d}-\text{DNNF} \) 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:

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

5 Conclusion

In this paper, the concept of tree-of-BDDs has been extended to any complete propositional representation language \( \text{C} \) thus leading to the family of \( \text{ToC} \) languages. A number of generic results are provided, which allow to determine the queries/transformations satisfied by \( \text{ToC} \) depending on the ones satisfied by \( \text{C} \), as well as results about the spatial efficiency of the \( \text{ToC} \) languages. Focusing on the \( \text{ToOBDD}_< \) 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 \( \text{CO} \) and \( \text{VA} \), \( \text{ToOBDD}_< \) satisfies \( \text{IM} \) and \( \text{ME} \) but does not satisfy any query among \( \text{CE}, \text{SE} \). We have also proved that \( \text{ToOBDD}_< \) does not satisfy any transformation among \( \text{CD}, \text{FO}, \land \text{BC}, \lor \text{C} \) or \( \lnot \text{C} \) and that this fragment is not comparable for succinctness w.r.t. any of \( \text{CNF}, \text{DNF}, \) and \( \text{DNNF} \) unless \( \text{PH} \) collapses.

From this investigation, it turns out that the \( \text{ToOBDD}_< \) language (and in general the \( \text{ToC} \) languages) satisfies only few queries and transformations. Subsequently, in applications where some queries/transformations not satisfied by \( \text{ToOBDD}_< \) must be achieved under some guaranteed response time, considering \( \text{ToOBDD}_< \) as a target language for \( \text{KC} \) is not always the best choice. However, imposing further restrictions on \( \text{C} \) can be a way to recover some of them. Thus, it is easy to show that if \( \text{C} \) has a linear-time algorithm for \( \text{FO} \) and a linear-time algorithm for \( \land \text{BC} \), then \( \text{ToC} \) has a polytime algorithm for \( \text{FO} \). If, in addition, \( \text{C} \) has a polytime algorithm for \( \text{CD} \), then \( \text{ToC} \) has a polytime algorithm for \( \text{CD} \). Furthermore, the fact that many queries/transformations are \( \text{NP} \)-hard in the general case does not discard \( \text{ToOBDD}_< \) (and beyond it the \( \text{ToC} \) languages) as interesting target languages for \( \text{KC} \) 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 ToC representation $T$, i.e. $\max_{n \in \mathbb{N}}(|\text{Var}(n)| - 1)$, is (upper) bounded by a constant,\footnote{At the language level, the price to be paid by such a restriction is a lack of expressiveness: none of the languages of ToC representations of width bounded by $c$ (where $c$ is a parameter) is complete (this is a direct consequence of Lemma 1).} then provided that the time complexities of forgetting variables in a C representation and conjoining two 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 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 ToOBDD $\nsubseteq$ CNF), there are CNF formulae which can be compiled into ToOBDD using a reasonable amount of computational resources, while it turned out impossible to generate $d$-DNNF representations for them. Such empirical results cohere with our succinctness result $d$-DNNF $\nsubseteq$ ToOBDD. Nevertheless, our result ToOBDD $\nsubseteq$ DNNF shows that this empirical evidence can be argued (this result implies that some DNNF representations do not have “small” ToOBDD equivalent representations under the standard assumptions of complexity theory), so DNNF remains a very attractive language for the KC purpose.

Our results also suggest a number of ToC languages as quite promising. Consider for instance the ToBFDD language. From our results, it comes easily that ToBFDD satisfies CO, VA, IM, ME (hence the same queries as ToOBDD); since ToBFDD is at least as succinct as ToOBDD, it appears as a challenging fragment. Furthermore, a compiler to BFDD is already available (see e.g. www.eecg.utoronto.ca/~jzhu/fbdduser11.ps)

When none of VA or IM is expected, the ToDNNF language looks also valuable; indeed, from our results we know that ToDNNF satisfies CO and ME, while being quite compact: ToDNNF $\nsubseteq$ ToOBDD and ToDNNF $\nsubseteq$ DNNF hold; beyond the spatial dimension, targeting the 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 CNF-to-ToC compilation time) since DNNF satisfies FO, which is one of the two key operations of the propagation algorithm. The ToDNNF$_T$ language, based on DNNF$_T$ [Pipatsrisawat and Darwiche, 2008], also looks interesting in this respect since it satisfies both FO and $\wedge\text{BC}$, 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 ToC languages and to extend the KC map accordingly. Especially, it would be interesting to characterize some families of propositional formulae each of DNNF and 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).

References


Appendix

Proof:[Proposition 1]

1. Proposition 3 shows that $C \geq_{P} \text{Toc}$. Hence, it holds that if $\text{Toc}$ satisfies CO, then $C$ satisfies CO. Conversely, a formula $\Sigma$ is inconsistent iff whatever the set of variables $V$, we have that $\exists V. \Sigma$ is inconsistent. Accordingly, a $\text{Toc}$ representation represents an inconsistent formula $\Sigma$ 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 $C \geq_{P} \text{Toc}$ (Proposition 3), we get that if $\text{Toc}$ satisfies VA (resp. IM), then $C$ satisfies VA (resp. IM). Conversely, let $T$ be a $\text{Toc}$ representation of a formula $\Sigma$. 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 $\gamma$ implies $\Sigma(T)$ iff $\gamma$ 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 $C \geq_{P} \text{Toc}$ (Proposition 3), we get that if $\text{Toc}$ satisfies ME, then $C$ satisfies ME. Conversely, we can design an output-polynomial time algorithm for computing the models of a $\text{Toc}$ representation $T$ over $\text{Var}(T)$. This algorithm is a recursive algorithm taking as inputs a $\text{Toc}$ representation $T$ and a term $\gamma$ such as $\text{Var}(\gamma) \subseteq \text{Var}(n_0)$ (where $n_0$ is the root of $T$). Initially, $\gamma$ is the empty term and the current node $n \in N$. $B(n)$ is conditioned by $\gamma$ (in polynomial time, since $C$ satisfies CD). The algorithm then computes the models $\omega$ of the conditioning $B(n)|\gamma$ of $B(n)$ by $\gamma$ (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 $\omega$ $B(n)|\gamma$, the procedure is recursively called on each of the children of $n$ (which is itself the root of $\text{Toc}$ representations) with input $\gamma = \omega$. This returns as many sets of models $\Omega_i$ as the number of children of $n$. Thanks to the running intersection property, the cartesian product $P_\omega$ of these $\Omega_i$ together with $\{\omega\}$ is the set of models of $\Sigma(T_n)$ over $\text{Var}(T_n)$, 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 $\omega$, of the $P_\omega$.

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

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

Proof:[Proposition 2]

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

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

2. $\lor C$. When $\exists C$ satisfies $\lor C$, every term $\gamma$ has a $C$ representation of size polynomial in the size of $\gamma$ Since $\text{DNF}$ does not satisfy VA unless $P = NP$, if $C$ satisfies it, it cannot be the case that $C$ satisfies $\lor C$ unless $P = NP$.

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

Proof:[Proposition 3]

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

2. To every $\text{Toc'}$ representation $T'$ of a formula $C$, we can associate via a poly-size function a $\text{Toc}$ representation $T$ of $\Sigma$. 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 c', there exists a poly-size 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 T^C does not satisfy CE, unless P = NP when c satisfies CL (point 5. of Proposition 1). Let p be any non-negative integer. Let \( \Sigma_p^{\max} \) be the CNF formula

\[
\bigwedge_{\delta \in 3-C_p} \neg \text{holds}_i \lor \delta_i
\]

where 3 - C_n is the set of all 3-literal clauses that can be generated from \( \{x_1, \ldots, x_p\} \) and the \text{holds}_i are new variables, not among \( x_1, \ldots, x_p \). The size \( |\Sigma_p^{\max}| \) of \( \Sigma_p^{\max} \) is in \( O(p^3) \) since \( \Sigma_p^{\max} \) contains \( O(p^3) \) 4-literal clauses. Especially, the number k of clauses of \( \Sigma_p^{\max} \) is cubic in p. Now, we associate to \( \Sigma_p^{\max} \) in polynomial time the following T^C representation T of it: the set of nodes of T is \( N = \{n_0, n_1, \ldots, n_k\} \), where \( n_0 \) is the root of T and the \( k \) remaining nodes are the \( k \) children of \( n_0 \). Furthermore, we have \( \text{Var}(n_i) = \{x_1, \ldots, x_p\} \) and for every \( i = 1, \ldots, k \), \( \text{Var}(n_i) = \{\text{holds}_i\} \cup \text{Var}(\delta_i) \).

Now, by construction, every clause which is a logical consequence of \( \Sigma_p^{\max} \) contains one of the \( \neg \text{holds}_i \) as a literal. Hence forgetting every \text{holds}_i in \( \Sigma_p^{\max} \) leads to a valid formula, and forgetting all the \text{holds}_i in \( \Sigma_p^{\max} \) except one, say \text{holds}_j, leads to a formula equivalent to the clause \( \neg \text{holds}_j \lor \delta_j \). We can compute in time polynomial in the size of \( \Sigma_p^{\max} \) the remaining labels \( B(n_i) = 1 \) and \( B(n_i) = 1 \) for each \( i = 1, \ldots, k \) as the \text{representation} of the clause \( \neg \text{holds}_i \lor \delta_i \), since c satisfies CL.

Each 3-CNF formula \( \alpha_p \), built up from the set of variables \( \{x_1, \ldots, x_p\} \) is in bijection with the subset \( S_{\alpha_p} \) of the variables \text{holds}_i s.t. \( \delta_i \) is a clause of \( \alpha_p \) if and only if \( \text{holds}_i \in S_{\alpha_p} \) (each variable \text{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 \( \delta_{\alpha_p} \) be the clause

\[
\bigvee_{\text{holds}_i \in S_{\alpha_p}} \neg \text{holds}_i.
\]

It is easy to check that \( \alpha_p \) is inconsistent iff \( \Sigma_p^{\max} \models \neg \delta_{\alpha_p} \).

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

\[
A(n) = \text{comp}(T).
\]

Once this is done, it determines whether \( \delta_{\alpha} \) is entailed by \( \text{comp}(T) \), which is in P. Since 3-SAT is complete for NP, this would imply NP \( \subseteq \text{P/poly} \), 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 \( \Sigma_p^{\max} \) be the CNF formula

\[
\bigwedge_{\delta \in 3-C_p} \neg \text{holds}_i \lor \delta_i
\]

where 3 - C_n is the set of all 3-literal clauses that can be generated from \( \{x_1, \ldots, x_p\} \) and the \text{holds}_i are new variables, not among \( x_1, \ldots, x_p \). The size \( |\Sigma_p^{\max}| \) of \( \Sigma_p^{\max} \) is in \( O(p^3) \) since \( \Sigma_p^{\max} \) contains \( O(p^3) \) 4-literal clauses. Especially, the number k of clauses of \( \Sigma_p^{\max} \) is cubic in p. Obviously enough, each 3-CNF formula \( \alpha_p \) built up from the set of variables \( \{x_1, \ldots, x_p\} \) in bijection with the subset \( S_{\alpha_p} \) of the variables \text{holds}_i s.t. \( \delta_i \) is a clause of \( \alpha_p \), if and only if \text{holds}_i \in S_{\alpha_p} \) (each variable \text{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 \( \delta_{\alpha_p} \) be the clause

\[
\bigvee_{\text{holds}_i \in S_{\alpha_p}} \neg \text{holds}_i.
\]

It is easy to check that \( \alpha_p \) is inconsistent if and only if \( \Sigma_p^{\max} \models \neg \delta_{\alpha_p} \).

Suppose now that T^C is at least at succinct as DNF. Then using a poly-size compilation function \text{comp} we could associate to DNF a \( \neg \Sigma_p^{\max} \) an equivalent T^C representation T. If c satisfies IM, then from point 3. of Proposition 1, T^C satisfies IM as well. So, for every term \( \gamma \), determining whether \( \gamma \) is an implicit 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 \( \alpha \) is satisfiable using a deterministic Turing machine with a polynomial advice A: if \( |\text{Var}(\alpha)| = p \), then the machine loads

\[
A(n) = T.
\]

Once this is done, it determines whether \( \gamma_{\alpha} = \neg \delta_{\alpha} \) is an implicit of T, which is in P. Since 3-SAT is complete for NP, this would imply NP \( \subseteq \text{P/poly} \), 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 \( \Sigma \) such that for each \( i \in 1 \ldots k \), \( \text{Var}(\delta) \notin V_i \). Since for each \( i \in 1 \ldots k \),

\[
\text{PID}(\exists V_i, \Sigma) = \{\delta' \in \text{PID}(\Sigma) \mid \text{Var}(\delta') \subseteq V_i\},
\]

we have that

\[
\delta \notin \bigcup_{i=1}^k \text{PID}(\exists V_i, \Sigma).
\]

Since \( \bigcup_{i=1}^k \text{PID}(\exists V_i, \Sigma) \) is a subset of \( \text{PID}(\Sigma) \) not containing \( \delta \) and since the conjunction of all prime implicates of \( \Sigma \) except \( \delta \) does not imply \( \delta \), by monotony of \( \models \), we get that

\[
\bigwedge_{i=1}^k \text{PID}(\exists V_i, \Sigma) \not\models \delta.
\]

However,

\[
\bigwedge_{i=1}^k \text{PID}(\exists V_i, \Sigma) \equiv \Sigma
\]
whenever \(\{V_1, \ldots, V_k\}\) is a decomposition set for \(\Sigma\), so we must also have

\[
\bigwedge_{i=1}^{k} PI(\exists \forall_i \Sigma) \models \delta
\]

since \(\delta\) is an implicate of \(\Sigma\), contradiction. \(\blacksquare\)

**Proof:** [Proposition 4]

- **Queries.**

  - As to CO, VA, IM, and ME, given the inclusions \(\text{ToOBDD}_\prec \subseteq \text{U(ToOBDD}_\prec \) \(\subseteq \text{ToOBDD}\), 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 \(\text{ToOBDD}_\prec \subseteq \text{U(ToOBDD}_\prec \) \(\subseteq \text{ToOBDD}\), it is enough to consider the case of ToOBDD representations. Since ToOBDD satisfies CL, points 5. and 6. of Proposition 1 concludes the proof.

- **Transformations.**

  - CD. Given the inclusions \(\text{ToOBDD}_\prec \subseteq \text{U(ToOBDD}_\prec \) \(\subseteq \text{ToOBDD}\), it is enough to consider the case of ToOBDD representations. Since OBDD satisfies CO and CL, Corollary 1 gives the result.

  - FO. None of ToOBDD \(\prec\) (whatever \(\prec\)), ToOBDD or U(ToOBDD) satisfies FO unless PH collapses. The proof is similar to the one used to show that OBDD \(\prec\) 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 ToOBDD representation \(T\) such that the set of nodes of \(T\) is \(\{n_0\}\) and \(B(n_0)\) is the OBDD formula equivalent to \(\Delta^1\) inductively defined by:

    * \(\Delta^k = \text{OBDD}_\prec(\gamma_k)\), i.e. the OBDD formula equivalent to the term \(\gamma_k\).

    * \(\Delta^1\) is the OBDD of the form \((\text{new}_i \land \text{OBDD}_\prec(\gamma_i)) \lor (\neg\text{new}_i \land \Delta^1)\) for each \(i \in 1 \ldots k - 1\).

Here each \(\text{new}_i\) \((i \in 1 \ldots k)\) is a fresh variable not occurring in \(\Sigma\). We assume that the \(\text{new}_i\) variables \((i \in 1 \ldots k)\) are earlier than all other variables w.r.t. \(\prec\). We have \(\Sigma \equiv \bigvee\{\text{new}_1, \ldots, \text{new}_k\}.\Sigma(T)\). Obviously, \(T\) also is a ToOBDD representation of \(\Sigma\) and a U(ToOBDD) representation of \(\Sigma\). If ToOBDD \(\prec\) (resp. ToOBDD, U(ToOBDD)) would satisfy FO, then we could turn in polynomial time the DNF formula \(\Sigma\) into an equivalent ToOBDD \(\prec\) (resp. ToOBDD, U(ToOBDD)) representation \(\hat{T}\). Given the inclusions \(\text{ToOBDD}_\prec \subseteq \text{U(ToOBDD}_\prec \) \(\subseteq \text{ToOBDD}\), since \(\text{U(ToOBDD}_\prec \) \(\not\subseteq\) DNF unless PH collapses, this would make PH to collapse.

- \(\wedge BC\). Each of ToOBDD \(\prec\) (whatever \(\prec\)), ToOBDD and U(ToOBDD) satisfies CO (see above) and TE (since this is the case for OBDD). but does not satisfy CE unless \(P = NP\) (see above). Then point 1. of Proposition 2 achieves the proof.

- \(\vee BC\). It is known that checking whether the conjunction of two OBDD \(\prec\) formulae \(\alpha\) and \(\beta\) (w.r.t. two different variable orderings \(\prec\)) is consistent is NP-complete (see Lemma 8.14 in [Meinel and Theobold, 1998]). Given that each OBDD \(\prec\) language (whatever \(\prec\)) satisfies \(\neg\text{C}\), unless \(P = NP\), there is no polynomial-time algorithm for deciding whether the disjunction of two OBDD \(\prec\) formulae (w.r.t. two different variable orderings \(\prec\)) is valid. Now, every OBDD \(\prec\) language (whatever \(\prec\)) is polynomially translatable into TOOBDD and into U(ToOBDD), given that ToOBDD \(\subseteq U(ToOBDD)\). Since each of ToOBDD and U(ToOBDD) satisfies VA, none of them can satisfy \(\vee\text{BC}\) unless \(P = NP\).

- \(\neg\text{C}\). Each of ToOBDD \(\prec\) (whatever \(\prec\)), ToOBDD and U(ToOBDD) satisfies IM (see above) but does not satisfy CE unless \(P = NP\) (see above). Then point 3. of Proposition 2 achieves the proof. \(\blacksquare\)

**Proof:** [Proposition 5]

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

4. Consider the formula

\[
\Sigma = \bigwedge_{i=1}^{m} \bigvee_{j=1}^{m} \neg x_{i,j} \land \bigwedge_{i=1}^{m} \bigvee_{j=1}^{m} \neg x_{i,j} \land \bigvee_{i=1}^{m} \bigwedge_{j=1}^{m} x_{i,j}
\]

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

\[
\bigwedge_{i=1}^{m} \bigvee_{j=1}^{m} \neg x_{i,j} \land \bigwedge_{i=1}^{m} \bigvee_{j=1}^{m} \neg x_{i,j}
\]

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

\[ B(n) \equiv \exists \text{Var}(\Sigma), \Sigma \equiv \Sigma \]

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

\[
(\bigwedge_{i=1}^{m} (\bigvee_{j=1}^{m} \neg x_{i,j})) \land (\bigwedge_{j=1}^{m} (\bigvee_{i=1}^{m} \neg x_{i,j}))
\]

has a size in \( \Omega(2^{\frac{m}{\sqrt{2}}}) \) (the proof of [Horiyama and Ibaraki, 2002] still applies to our formula \( \Sigma \) since the proposed fooling set \( A \) containing \( 2^{\frac{m}{\sqrt{2}}} \) assignments also is a fooling set when \( \Sigma \) is considered instead of

\[
(\bigwedge_{i=1}^{m} (\bigvee_{j=1}^{m} \neg x_{i,j})) \land (\bigwedge_{j=1}^{m} (\bigvee_{i=1}^{m} \neg x_{i,j})).
\]