Journal on Satisfiability, Boolean Modeling and Computation 1 (2006) 61–88

Complexity Results for Quantified Boolean Formulae Based on Complete Propositional Languages*

Sylvie Coste-Marquis {coste,leberre,letombe,marquis}@cril.univ-artois.fr
Daniel Le Berre
Florian Letombe
Pierre Marquis
CRIL/CNRS, Université d'Artois,
rue de l'Université — S.P. 16,
F-62307 Lens, France

Abstract

Several propositional fragments have been considered so far as target languages for knowledge compilation and used for improving computational tasks from major AI areas (like inference, diagnosis and planning); among them are the ordered binary decision diagrams, prime implicates, prime implicants, "formulae" in decomposable negation normal form. On the other hand, the validity problem VAL($\mathtt{QPROP}_{PS}$) for Quantified Boolean Formulae (QBF) has been acknowledged for the past few years as an important issue for AI, and many solvers have been designed. In this paper, the complexity of restrictions of the validity problem for QBF obtained by imposing the matrix of the input QBF to belong to propositional fragments used as target languages for compilation, is identified. It turns out that this problem remains hard (PSPACE-complete) even under severe restrictions on the matrix of the input. Nevertheless some tractable restrictions are pointed out.

KEYWORDS: *automated reasoning, quantified Boolean formulae*

Submitted September 2005; revised March 2006; published March 2006

1. Introduction

Compiling "knowledge" has been used for the past few years to improve (from the computational point of view) some basic tasks from major AI areas, like inference (both classical and nonmonotonic, see among others [1, 2, 3, 4, 5, 6, 7]), diagnosis (see e.g. [8, 9]) and planning (see e.g. [10, 11, 12]). These approaches typically consist in turning, during an off-line phase, some pieces of information encoded as propositional formulae into formulae from a "more tractable" fragment $\mathcal{C}$. "More tractable" means that the tasks required by the application under consideration become computationally easier, and if possible, feasible in polynomial time when the input belongs to such a fragment [13]. Such tasks usually contain deciding satisfiability (determining whether a given propositional formula has or not a model), the famous SAT problem, which is NP-complete. Among the "tractable" frag-

* A preliminary version of this paper appeared in the proceedings of the 20^{th} National Conference on Artificial Intelligence (AAAI'05), pp. 288-293.

ments considered so far are the (quite influential) ordered binary decision diagrams, prime implicates, prime implicants, and "formulae" in decomposable negation normal form.

On the other hand, VAL($\mathtt{QPROP}_{PS}$), the validity problem for QBF, has a growing importance in AI. This can be explained by the fact that, as the canonical PSPACE-complete problem, many AI problems can be polynomially reduced to VAL($\mathtt{QPROP}_{PS}$) (see e.g., [14, 15, 16, 17, 18]); in particular, VAL($\mathtt{QPROP}_{PS}$) includes SAT as a specific case; furthermore, there is some empirical evidence from various AI fields (including among others planning, nonmonotonic reasoning, paraconsistent inference) that a translation-based approach can prove more "efficient" than domain-dependent algorithms dedicated to such AI tasks. Accordingly, many solvers for VAL($\mathtt{QPROP}_{PS}$) have been designed and evaluated for the past few years (see among others [19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29]).

In this paper, we consider several tractable fragments for SAT, used as target languages for knowledge compilation. For each fragment $\mathcal{C}$ under consideration, we focus on the restriction VAL($\mathtt{Q}\mathcal{C}$) of the VAL($\mathtt{QPROP}_{PS}$) problem obtained by imposing the matrix of the input formula to belong to the fragment.

A similar investigation has already been done w.r.t. some *incomplete* propositional fragments [30, 31]. Thus, in his well-known paper where a dichotomy theorem for SAT is presented [30], Schaefer also gave an analogue dichotomy theorem for VAL($\mathtt{QPROP}_{PS}$) (Theorem 6.1); roughly, this theorem shows that the only tractable classes for the restrictions of VAL($\mathtt{QPROP}_{PS}$) among those "characterized locally" (i.e., by the nature of the "clauses" from the matrix) are the Krom one (binary clauses), the Horn one, the reverse Horn one and the affine one (sets of linear equations over the field $\{0, 1\}$, or equivalently, conjunctions of XOR-clauses). Accordingly, several polytime algorithms for the restriction of VAL($\mathtt{QPROP}_{PS}$) to such incomplete fragments can be found in the literature (see [32, 33, 34]).

In the very recent past, VAL($\mathtt{QCSP}$), the validity problem for quantified constraint networks – a generalization of VAL($\mathtt{QPROP}_{PS}$) when conjunctive constraints are considered – has received much attention; classification theorems giving the complexity of VAL($\mathtt{QCSP}$) depending on algebraic properties of the constraint language under consideration have been pointed out [35, 36, 37]; such impressive results are deep insights into the study of the complexity of VAL($\mathtt{QCSP}$); nevertheless, they are concerned in essence with conjunctive constraints, so they cannot be used directly as such to identify the complexity of every restriction of VAL($\mathtt{QCSP}$) (just like Schaefer's dichotomy theorem for SAT does not characterize every tractable restriction of SAT, like the ones based on ordered binary decision diagrams or on renamable Horn CNF formulae).

In this paper, the complexity of VAL($\mathtt{Q}\mathcal{C}$) is investigated for *complete* propositional fragments $\mathcal{C}$, where a propositional fragment $\mathcal{C}$ is complete if and only if every propositional formula has an equivalent into $\mathcal{C}$. We mainly focus on fragments $\mathcal{C}$ considered in [13]: $\mathtt{DNF}$, $\mathtt{d\text{-}DNNF}$, $\mathtt{sd\text{-}DNNF}$, $\mathtt{DNNF}$, $\mathtt{OBDD}_<$, $\mathtt{FBDD}$, $\mathtt{PI}$, $\mathtt{IP}$, $\mathtt{MODS}$ whose significance for many AI tasks (as well as for problems pertaining to other fields) is acknowledged. We complete the results given in [13] by focusing on an additional query, the VAL($\mathtt{Q}\mathcal{C}$) one. We draw the complexity picture for VAL($\mathtt{Q}\mathcal{C}$) for all those fragments $\mathcal{C}$. We also consider the VAL($\mathtt{Q}\mathcal{C}$) for two additional fragments: $\mathtt{OCNF}_<$and $\mathtt{ODNF}_<$.

Both tractability and intractability results have been derived. Like for the $\mathtt{DNF}$ fragment and its supersets including the $\mathtt{DNNF}$ fragment and the disjunctions of Horn $\mathtt{CNF}$formulae, the VAL($\mathtt{Q}\mathcal{C}$) problem for the $\mathcal{C} = \mathtt{OBDD}_<$ fragment (and its supersets, the $\mathtt{FBDD}$ fragment

and the d-DNNF one) is PSPACE-complete in the general case, while in P whenever the prefix of the instance is compatible with the total, strict ordering $<$ associated with the $\mathtt{OBDD}_<$ "formula". We also consider the complete restrictions $\mathtt{ODNF}_<$ of DNF (and dually, the restrictions $\mathtt{OCNF}_<$ of CNF) and show that for those two sets of fragments $\mathcal{C}$ ($<$ varying), $\textsc{val}(\mathtt{Q}\mathcal{C})$ is in P under the compatibility assumption. Because QMODS is a subset of $\mathtt{QODNF}_<$ for which the compatibility assumption holds (and dually, the fragments QCI of quantified canonical implicates formulae is a subset of $\mathtt{QOCNF}_<$ for which the compatibility assumption holds), we get that the $\textsc{val}(\mathtt{QMODS})$ problem and the $\textsc{val}(\mathtt{QCI})$ problem are in P as well. We finally show that the $\textsc{val}(\mathtt{QPI})$ problem (resp. the $\textsc{val}(\mathtt{QIP})$ problem) is PSPACE-complete, while the complexity falls down to P for the restriction where the prefix of any instance is of the form $\forall X\ \exists Y$ (resp. $\exists X\ \forall Y$).

The rest of the paper is organized as follows: in Section 2, we give some formal preliminaries. In Section 3, the complexity results are presented. In Section 4, some experimental results are provided. In Section 5, the connection between the problem of finding out tractable matrix-based restrictions of $\textsc{val}(\mathtt{QPROP}_{PS})$ and the compilability issue for $\textsc{val}(\mathtt{QPROP}_{PS})$ when matrices are fixed while prefixes may vary is investigated. Finally, Section 6 concludes the paper and gives a few perspectives. We assume the reader familiar with classes of the polynomial hierarchy PH and the complexity class PSPACE, and with standard polynomial many-one reductions (see e.g. [38]).

2. Formal Preliminaries

In this section, we present the syntax and semantics of quantified boolean formulae; we also give a number of easy metatheorems which will prove useful in the following sections. We set the morphology of our language to the following set of connectives: $\neg$, $\wedge$, $\vee$, $\Rightarrow$, $\Leftrightarrow$, $\oplus$ (XOR) plus the two boolean constants and the quantifiers $\forall$ and $\exists$; on this ground, a language for quantified boolean formulae can be defined as follows:

Definition 1 (syntax of a quantified boolean formula)**.** *Let PS be a finite set of propositional symbols. The set $\mathtt{QPROP}_{PS}$ of* quantified boolean formulae *(QBFs) over PS is the smallest set of words defined inductively as follows:*[1.]

1. *the boolean constants true, false and every variable from PS belong to $\mathtt{QPROP}_{PS}$.*
2. *if ϕ and ψ belong to $\mathtt{QPROP}_{PS}$ then $(\neg\phi)$, $(\phi \wedge \psi)$, $(\phi \vee \psi)$, $(\phi \Rightarrow \psi)$, $(\phi \Leftrightarrow \psi)$, $(\phi \oplus \psi)$ belong to $\mathtt{QPROP}_{PS}$.*
3. *if ϕ belongs to $\mathtt{QPROP}_{PS}$ and x belongs to PS, then $(\forall x.\phi)$ and $(\exists x.\phi)$ belong to $\mathtt{QPROP}_{PS}$.*
4. *every quantified propositional formula is obtained by applying the three rules above a finite number of times.*

Example 1. *The following formula is a QBF:*

$$\Sigma = \exists a.(((\forall b.(a \wedge b)) \vee ((\neg a) \vee b)) \wedge \forall b.(a \vee ((\neg b) \wedge c)))$$

Figure 1 is a graphical representation of Σ.

1. In order to simplify the syntax, we feel free to omit some parentheses when this does not question equivalence. In addition, we often abbreviate $\exists x.(\exists y.\phi)$ (resp. $\forall x.(\forall y.\phi)$) into $\exists x, y.\phi$ (resp. $\forall x, y.\phi$).

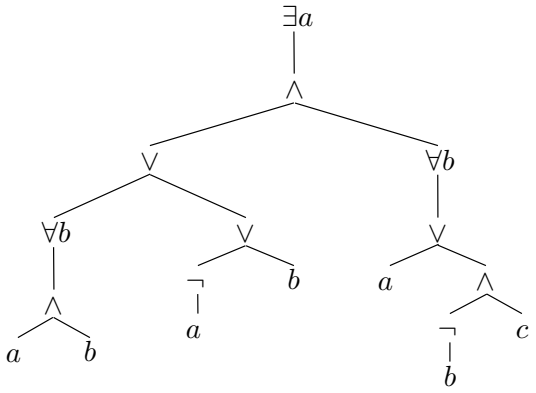


Figure 1. A quantified boolean formula.

■

The occurrences of a propositional variable x in a formula Σ from $\texttt{QPROP}_{PS}$ can be partitioned into three sets: the *quantified* occurrences of x in Σ are those occurring in a quantification, i.e., just after a quantifier $\forall$ or $\exists$; in every subformula $\forall x.\phi$ (resp. $\exists x.\phi$) of Σ, all the occurrences of x in ϕ are *bound*, the occurrences of x are said to be *in the scope of the quantification* $\forall x$ (resp. $\exists x$). Finally, all the remaining occurrences of x in Σ are *free* ones.

$Var(\Sigma)$ is the set of all variables occurring in Σ. A variable x of Σ is *free* if it has a free occurrence in Σ.

Example 2 (cont'ed)**.** *The first occurrence of a in Σ is quantified, the second occurrence of b in Σ is bound and the third one is free. b and c are the free variables of Σ.* ■

A QBF Σ is said to be *polite* if and only if every bound occurrence of a variable x of Σ is in the scope of an unique quantification, and every free variable has no bound occurrence. A QBF Σ is said to be *prenex* if and only if $\Sigma = Qx_1 \ldots Qx_n.\phi$ where each occurrence of Q stands for either $\forall$ or $\exists$, and ϕ does not contain any quantified occurrence of a variable. ϕ is said to be the *matrix* of Σ and the sequence $Qx_1 \ldots Qx_n$ of quantifications is the *prefix* of Σ. A QBF is said to be *closed* if and only if it has no free variable. A QBF is said to be *quantifier-free* if and only if it does not contain any quantification (obviously enough, such formulae can easily be considered as "standard" propositional formulae). The subset of $\texttt{QPROP}_{PS}$ containing only quantifier-free formulae is noted $\texttt{PROP}_{PS}$.

Example 3 (cont'ed)**.** *Σ is neither polite, nor prenex, nor closed.* ■

Let us now consider the semantical aspects of QBF; let us start with the following useful notion of conditioning (also referred to as restriction or cofactor [39]). For every quantified boolean formula Σ and every variable x, $\Sigma_{x \leftarrow 0}$ (resp. $\Sigma_{x \leftarrow 1}$) denotes the QBF obtained by replacing every free occurrence of x in Σ by $false$ (resp. $true$). Formally:

Definition 2 (conditioning). *Let $\Sigma \in \mathtt{QPROP}_{PS}$, $x \in PS$ and $* \in \{1, 0\}$. The* conditioning *of x by $*$ in Σ is the QBF defined inductively as follows:*

$$\Sigma_{x \leftarrow *} = \begin{cases} if\ \Sigma = x\ and\ * = 1\ then\ true \\ if\ \Sigma = x\ and\ * = 0\ then\ false \\ if\ \Sigma \in PS\ and\ \Sigma \neq x\ then\ \Sigma \\ if\ \Sigma = \neg\phi\ then\ \neg(\phi_{x \leftarrow *}) \\ if\ \Sigma = \phi \circ \psi\ then\ \phi_{x \leftarrow *} \circ \psi_{x \leftarrow *}\ with\ \circ\ a\ binary\ connective \\ if\ \Sigma = \forall x.\phi\ or\ \Sigma = \exists x.\phi\ then\ \Sigma \\ if\ \Sigma = \forall y.\phi\ with\ y \neq x\ then\ \forall y.(\phi_{x \leftarrow *}) \\ if\ \Sigma = \exists y.\phi\ with\ y \neq x\ then\ \exists y.(\phi_{x \leftarrow *}) \end{cases}$$

Example 4 (cont'ed). *The conditioning of b by 1 in Σ is the QBF*

$$\Sigma_{b \leftarrow 1} = \exists a.(((\forall b.(a \wedge b)) \vee ((\neg a) \vee true)) \wedge \forall b.(a \vee ((\neg b) \wedge c)))$$

■

We are now ready to define the semantics of a QBF:

Definition 3 (semantics of a quantified boolean formula). *Let I be an interpretation over PS (i.e., a total function from PS to $BOOL = \{0, 1\}$). The* semantics *of a quantified boolean formula Σ in I is the truth value $[\![\Sigma]\!](I)$ from $BOOL$ defined inductively as follows:*

- *if $\Sigma = true$ (resp. $false$), then $[\![\Sigma]\!](I) = 1$ (resp. 0).*
- *if $\Sigma \in PS$, then $[\![\Sigma]\!](I) = I(\Sigma)$.*
- *if $\Sigma = \neg\phi$, then $[\![\Sigma]\!](I) = 1 - [\![\phi]\!](I)$.*
- *if $\Sigma = \phi \wedge \psi$, then $[\![\Sigma]\!](I) = [\![\phi]\!](I) \times [\![\psi]\!](I)$.*
- *if $\Sigma = \phi \vee \psi$, then $[\![\Sigma]\!](I) = max(\{[\![\phi]\!](I), [\![\psi]\!](I)\})$.*
- *if $\Sigma = \phi \Rightarrow \psi$, then $[\![\Sigma]\!](I) = [\![\neg\phi \vee \psi]\!](I)$.*
- *if $\Sigma = \phi \Leftrightarrow \psi$, then $[\![\Sigma]\!](I) = [\![(\phi \Rightarrow \psi) \wedge (\psi \Rightarrow \phi)]\!](I)$.*
- *if $\Sigma = \phi \oplus \psi$, then $[\![\Sigma]\!](I) = [\![\neg(\phi \Leftrightarrow \psi)]\!](I)$.*
- *if $\Sigma = \forall x.\phi$, then $[\![\Sigma]\!](I) = min(\{[\![\phi_{x \leftarrow 0}]\!](I), [\![\phi_{x \leftarrow 1}]\!](I)\})$.*
- *if $\Sigma = \exists x.\phi$, then $[\![\Sigma]\!](I) = max(\{[\![\phi_{x \leftarrow 0}]\!](I), [\![\phi_{x \leftarrow 1}]\!](I)\})$.*

Clearly enough, every connective used in the morphology of $\mathtt{QPROP}_{PS}$ (including quantifications which can be viewed as unary connectives) is truth-functional.

An interpretation I is said to be a *model* of Σ, noted $I \models \Sigma$, if and only if $[\![\Sigma]\!](I) = 1$. If Σ has a model, it is *satisfiable*; otherwise, it is *unsatisfiable*. If every interpretation I over PS is a model of Σ, Σ is *valid*, noted $\models \Sigma$. If every model of Σ is a model of μ, then μ is a *logical consequence* of Σ, noted $\Sigma \models \mu$. Finally, when both $\Sigma \models \mu$ and $\mu \models \Sigma$ hold, Σ and μ are *equivalent*, noted $\Sigma \equiv \mu$.

It is not difficult to prove (again by structural induction) that the semantics of any quantified boolean formula Σ depends only on its free variables, in the sense that, for any interpretation J over PS which coincides with a given interpretation I on all the free variables of Σ, I is a model of Σ if and only if J is a model of Σ. Especially, the semantics of a closed formula is the same in every interpretation over PS. Stated otherwise, such a formula is equivalent to one of the boolean constants $true$ or $false$, hence it is satisfiable if and only if it is valid.

As in propositional logic, $\Sigma \models \mu$ holds if and only if the formula $(\Sigma \wedge \neg\mu)$ is unsatisfiable if and only if the formula $(\Sigma \Rightarrow \mu)$ is valid. More generally, since the connectives are truth-functional ones, a substitution metatheorem holds for quantified boolean formulae: replacing any subformula by an equivalent one preserves equivalence w.r.t. the overall formula.

It is easy to show that the conditioning of x by $*$ in Σ can be computed in time linear in $|\Sigma|$. Furthermore, it is easy to show by structural induction that successive conditionings commute: for any $\Sigma \in \mathtt{QPROP}_{PS}$, any $x, x' \in PS$ and $*, *' \in \{0,1\}$, we have

$$(\Sigma_{x \leftarrow *})_{x' \leftarrow *'} \equiv (\Sigma_{x' \leftarrow *'})_{x \leftarrow *}.$$

Other interesting metatheorems are given in the following two propositions.

Proposition 1 (folklore)**.** *Let Σ, Φ be formulae from $\mathtt{QPROP}_{PS}$ and x, y be variables from PS.*

1. $\forall x.\Sigma \equiv \Sigma_{x \leftarrow 0} \wedge \Sigma_{x \leftarrow 1}$.
2. $\exists x.\Sigma \equiv \Sigma_{x \leftarrow 0} \vee \Sigma_{x \leftarrow 1}$.
3. $\forall x.\Sigma \equiv \neg(\exists x.(\neg\Sigma))$.
4. *If x is not free in Σ, then* $\forall x.\Sigma \equiv \exists x.\Sigma \equiv \Sigma$.
5. $\forall x.(\Sigma \wedge \Phi) \equiv (\forall x.\Sigma) \wedge (\forall x.\Phi)$.
6. $\exists x.(\Sigma \vee \Phi) \equiv (\exists x.\Sigma) \vee (\exists x.\Phi)$.
7. $\forall x.(\forall y.\Sigma) \equiv \forall y.(\forall x.\Sigma)$.
8. $\exists x.(\exists y.\Sigma) \equiv \exists y.(\exists x.\Sigma)$.
9. *If x is not free in Σ, then* $\forall x.(\Sigma \vee \Phi) \equiv \Sigma \vee (\forall x.\Phi)$.
10. *If x is not free in Σ, then* $\exists x.(\Sigma \wedge \Phi) \equiv \Sigma \wedge (\exists x.\Phi)$.

Points 1 and 2 show that every quantified boolean formula Σ can be turned into a "standard" propositional formula but the transformation suggested (viewing the equivalences as left-to-right rewriting rules) cannot be achieved in polynomial space (and more generally, it is very unlikely that a polysize mapping from $\mathtt{QPROP}_{PS}$ to $\mathtt{PROP}_{PS}$ that preserves equivalence exist, since this would make the polynomial hierarchy PH to collapse). Accordingly, it is assumed that $\mathtt{QPROP}_{PS}$ enables much more compact encodings than $\mathtt{PROP}_{PS}$.

Point 3 shows that universal quantifiers and existential quantifiers are dual ones.

Point 4 gives a sufficient condition for the elimination of quantifications (the condition is not necessary as the example $\Sigma = x \vee \neg x$ shows it).

Points 5, 6, 9 and 10 make precise the interplay between the quantifiers and the connectives $\wedge$ and $\vee$.

Points 7 and 8 show that it is possible to switch two successive quantifications of the same nature in a quantified boolean formula Σ without questioning equivalence. Hence, for every finite, non empty subset $S = \{x_1, \ldots, x_n\}$ of PS, we note $\forall S.\Sigma$ (resp. $\exists S.\Sigma$) as a shortand for $\forall x_1, \ldots, \forall x_n.\Sigma$ (resp. $\exists x_1, \ldots, \exists x_n.\Sigma$).

Contrastingly, it is not possible in general to switch two successive quantifications of different nature while preserving equivalence. Thus, $\forall x.(\exists y.\Sigma)$ is a logical consequence of $\exists y.(\forall x.\Sigma)$ but is not equivalent to it (just consider $\Sigma = x \Leftrightarrow y$). Furthermore, in the general case, we do have neither $\forall x.(\Sigma \vee \Phi) \equiv (\forall x.\Sigma) \vee (\forall x.\Phi)$ nor $\exists x.(\Sigma \wedge \Phi) \equiv (\exists x.\Sigma) \wedge (\exists x.\Phi)$ (as a counterexample, take $\Sigma = x$ and $\Phi = \neg x$).

Based on the metatheorems given in Proposition 1 and the substitution metatheorem, it is easy to prove that every formula from $\texttt{QPROP}_{PS}$ can be turned in polynomial time into a prenex and polite, equivalent formula (bound variables can be renamed without questioning equivalence). Note that several strategies for prenexing a quantified boolean formula exist (depending on the way quantifications are shifted), and that the choice of a prenex formula equivalent to a given quantified boolean formula Σ when several are possible may have a practical impact on the efficiency of deciding the validity of Σ [40].

Example 5 (cont'ed)**.** *Σ is equivalent to the following prenex, polite QBF:*

$$\exists a.(\forall u.(\forall v.(((a \wedge u) \vee ((\neg a) \vee b)) \wedge (a \vee ((\neg v) \wedge c))))).$$

u and v are the fresh variables used to make the formula polite. ■

Finally, there are close connections between (general) quantified boolean formulae and closed ones. Especially we have:

Proposition 2 (folklore)**.** *Let Σ be a formula from $\texttt{QPROP}_{PS}$.*

1. *Σ is satisfiable if and only if the closed formula $\exists Var(\Sigma).\Sigma$ is valid.*
2. *Σ is valid if and only if the closed formula $\forall Var(\Sigma).\Sigma$ is valid.*

Points 1 and 2 above show that the satisfiability problem (resp. the validity problem) of (general) quantified boolean formulae can be reduced in polynomial time to the validity problem of closed quantified boolean formulae. Especially, the satisfiability problem of a "standard" propositional formula Σ can be reduced in polynomial time to the validity problem of the closed quantified boolean formula $\exists Var(\Sigma).\Sigma$.

Example 6 (cont'ed)**.** *Σ is satisfiable since*

$$\exists a, b, c.(\exists a.(((\forall b.(a \wedge b)) \vee ((\neg a) \vee b)) \wedge \forall b.(a \vee ((\neg b) \wedge c))))$$

which can be simplified to

$$\exists b, c.(\exists a.(((\forall b.(a \wedge b)) \vee ((\neg a) \vee b)) \wedge \forall b.(a \vee ((\neg b) \wedge c))))$$

since a is not free in Σ, is a valid, closed QBF. ■

For this reason, the VAL($\mathtt{QPROP}_{PS}$) problem is usually stated as follows:

Definition 4 (VAL($\mathtt{QPROP}_{PS}$))**.** VAL($\mathtt{QPROP}_{PS}$) *is the following decision problem:*

- **Input:** *A prenex, closed, polite formula* Σ *from* $\mathtt{QPROP}_{PS}$*;*
- **Question:** *Is* Σ *valid?*

Example 7 (cont'ed)**.** Σ *is satisfiable if and only if the following instance of* VAL($\mathtt{QPROP}_{PS}$) *is valid:*

$$\exists a, b, c\ \forall u, v.(((a \wedge u) \vee ((\neg a) \vee b)) \wedge (a \vee ((\neg v) \wedge c)))$$

■

More generally, we use the following notations:

Definition 5 (Notations)**.** *Let* $\mathcal{C} \subseteq \mathtt{PROP}_{PS}$*.*
We note:

- $\mathtt{Q}\mathcal{C}$ *is the language of prenex, closed, polite QBF with matrix from* $\mathcal{C}$*.*
- VAL($\mathtt{Q}\mathcal{C}$) *is the following decision problem:*
  - **Input:** *A formula* Σ *from* $\mathtt{Q}\mathcal{C}$*;*
  - **Question:** *Is* Σ *valid?*

3. Tractable vs. Intractable Classes for VAL($\mathtt{QPROP}_{PS}$)

In the following, the complexity of several restrictions of VAL($\mathtt{QPROP}_{PS}$) is investigated. A propositional fragment (i.e., a subset of a propositional language) $\mathcal{C}$ is said to be *tractable* for VAL($\mathtt{QPROP}_{PS}$) (resp. SAT) if and only if the membership to the fragment can be decided in polynomial time, and there also exists a polytime decision algorithm for the validity problem VAL($\mathtt{Q}\mathcal{C}$) (resp. there exists a polytime decision algorithm for the satisfiability problem for formulae from the fragment).

Let us start with intractability results. First, it is well-known that the restriction VAL($\mathtt{QCNF}$) of VAL($\mathtt{QPROP}_{PS}$) is still PSPACE-complete. Indeed, every propositional formula Σ over $\{x_1, \ldots, x_n\}$ can be associated in linear time to a CNF formula Σ' over $\{x_1, \ldots, x_n, y_1, \ldots, y_m\}$ s.t. $\Sigma \equiv \exists\{y_1, \ldots, y_m\}.\Sigma'$. Such a reduction which preserves satisfiability (and much more) is typically used to show that CIRCUIT-SAT can be reduced to SAT restricted to CNF formulae (the basic idea is to introduce a new variable y_i per gate or subformula).

Let us now consider the VAL($\mathtt{Q}\mathcal{C}$) problem for target fragments $\mathcal{C}$ for knowledge compilation; many such fragments have been identified in the literature: DNF, sd-DNNF, d-DNNF, DNNF, FBDD, $\mathtt{OBDD}_<$, MODS, PI, IP, ... As we will see, all such problems are typically intractable.

First, since PSPACE is closed under complementation and the negation of a QBF with a CNF matrix is a QBF with a DNF matrix, it follows directly that the VAL($\mathtt{QDNF}$) problem also is PSPACE-complete. This prevents many tractable fragments $\mathcal{C}$ for SAT from being considered as interesting candidates for VAL($\mathtt{Q}\mathcal{C}$). Among them are all the supersets of DNF

including the DNNF fragment and the disjunctions of Horn CNF formulae which are target classes for knowledge compilation (see [3, 4, 13]).

Let us now turn to complete DAG-based propositional fragments.[2] A "formula"[3] in $\mathtt{NNF}_{PS}$ is a rooted, directed acyclic graph where each leaf node is labeled with *true*, *false*, x or $\neg x$, $x \in PS$; and each internal node is labeled with $\wedge$ or $\vee$ and can have arbitrarily many children. If C is a node in an $\mathtt{NNF}_{PS}$ formula, then $Var(C)$ denotes the set of all variables that label the descendants of node C. Moreover, if ϕ is an $\mathtt{NNF}_{PS}$ formula rooted at C, then $Var(\phi)$ is defined as $Var(C)$. Interesting fragments of $\mathtt{NNF}_{PS}$ are obtained by imposing some of the following requirements [41]:

- **Decomposability**: An and-node C is decomposable if and only if the conjuncts of C do not share variables. That is, if $C_1, \ldots, C_n$ are the children of and-node C, then $Var(C_i) \cap Var(C_j) = \emptyset$ for $i \neq j$. An $\mathtt{NNF}_{PS}$ formula satisfies the decomposability property if and only if every and-node in it is decomposable.
- **Determinism**: An or-node C is deterministic if and only if each pair of disjuncts of C is logically contradictory. That is, if $C_1, \ldots, C_n$ are the children of or-node C, then $C_i \wedge C_j \models \mathit{false}$ for $i \neq j$. An $\mathtt{NNF}_{PS}$ formula satisfies the determinism property if and only if every or-node in it is deterministic.
- **Decision**: A decision node N in an $\mathtt{NNF}_{PS}$ formula is one which is labeled with *true*, *false*, or is an or-node having the form $(x \wedge \alpha) \vee (\neg x \wedge \beta)$, where x is a variable, α and β are decision nodes. In the latter case, $dVar(N)$ denotes the variable x. An $\mathtt{NNF}_{PS}$ formula satisfies the decision property when its root is a decision node.
- **Ordering**: Let $<$ be a total, strict ordering over the variables from PS. An $\mathtt{NNF}_{PS}$ formula satisfying the decision property satisfies the ordering property w.r.t. $<$ if and only if the following condition is satisfied: if N and M are or-nodes, and if N is an ancestor of node M, then $dVar(N) < dVar(M)$.
- **Smoothness**: An or-node C is smooth if and only if each disjunct of C mentions the same variables. That is, if $C_1, \ldots, C_n$ are the children of or-node C, then $Var(C_i) = Var(C_j)$ for $i \neq j$. An $\mathtt{NNF}_{PS}$ formula satisfies the smoothness property if and only if every or-node in it is smooth.

Example 8. *Consider the and-node marked ➘ on the left part of Figure 2. This and-node C has two children C_1 and C_2 such that $Var(C_1) = \{a, b\}$ and $Var(C_2) = \{c, d\}$; Node C is decomposable since the two children do not share variables. Each other and-node in Figure 2 (left) is also decomposable and, hence, the $\mathtt{NNF}_{PS}$ formula in this figure is decomposable. Consider now the or-node marked ➘ on the right part of the figure; it has two children corresponding to subformulae $\neg a \wedge b$ and $\neg b \wedge a$. Those two subformulae are jointly unsatisfiable, hence the or-node is deterministic. Furthermore, the two children*

2. Actually, the DAG-based fragment corresponding to CNF (resp. DNF) can be identified to the fragment CNF (resp. DNF) consisting of tree-like formulae – as defined before – without any significant loss w.r.t. succinctness. See [13].
3. In the following, we will use the term formula to denote the DAG-based representation of that formula.

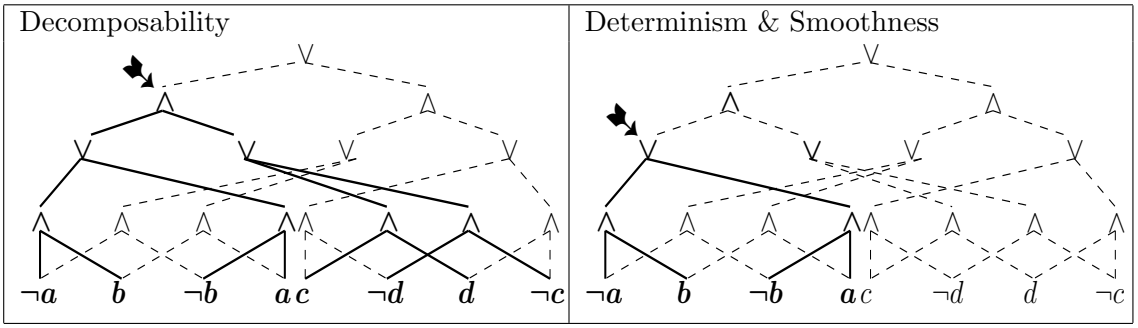


Figure 2. A formula in $\mathtt{NNF}_{PS}$. On the left part, the node marked ➘ is decomposable while the node marked with the same symbol on the right part denotes a deterministic and smooth node.

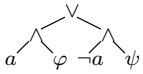
mention the same variables a and b, hence the or-node is smooth. Since the other or-nodes in Figure 2 (right) are also deterministic and smooth, the $\mathtt{NNF}_{PS}$ *formula in this figure is deterministic and smooth.* ■

We consider the following propositional fragments[4.] [13]:

Definition 6 (propositional fragments)**.**

- *The language* DNNF *is the subset of* $\mathtt{NNF}_{PS}$ *of formulae satisfying decomposability.*
- *The language* d-DNNF *is the subset of* $\mathtt{NNF}_{PS}$ *of formulae satisfying decomposability and determinism.*
- *The language* sd-DNNF *is the subset of* $\mathtt{NNF}_{PS}$ *of formulae satisfying decomposability, determinism and smoothness.*
- *The language* FBDD *is the subset of* $\mathtt{NNF}_{PS}$ *of formulae satisfying decomposability and decision.*
- *The language* $\mathtt{OBDD}_{<}$ *is the subset of* $\mathtt{NNF}_{PS}$ *of formulae satisfying decomposability, decision and ordering.*
- *The language* MODS *is the subset of* DNF ∩ d-DNNF *of formulae satisfying smoothness.*

The FBDD language corresponds to *free binary decision diagrams (FBDDs)*, as known in formal verification [42], while its subset obtained by imposing the ordering property w.r.t. a given variable ordering contains the *ordered binary decision diagrams (OBDDs)* [39].

Binary decision diagrams are usually depicted using a more compact notation: labels *true* and *false* are denoted by 1 and 0, respectively; and each decision node 

4. It must be noted that the six languages below are not *stricto sensu* subsets of $\mathtt{PROP}_{PS}$ in the sense that its elements are rooted DAGs, not standard tree-like formulae. Considering DAG-based representations is just a way to enable subformulae sharing; while this is fundamental for the spatial efficiency point of view, this has no impact on the semantical issue, so the definitions and properties reported in the Section 2 can be easily extended to DAG-based formulae.

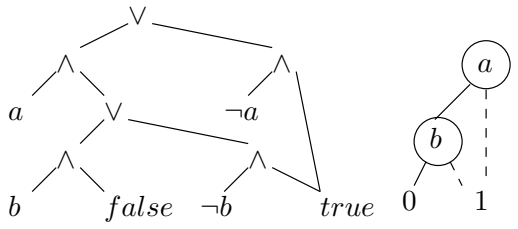


Figure 3. On the left part, a formula in the OBDD$_<$ language. On the right part, a more standard notation for it.

denoted by $\overset{a}{\underset{\varphi}{\diagdown}}\!\!\overset{}{\diagup}_{\psi}$. The OBDD$_<$ formula on the left part of Figure 3 corresponds to the binary decision diagram on the right part of Figure 3.

A MODS encoding of a propositional formula mainly consists of the explicit representation of the set of its models over the set of variables occurring in it. Figure 4 depicts a formula from MODS which is equivalent to the DNFformula $(a \wedge b \wedge c) \vee (\neg a \wedge \neg b \wedge c) \vee (\neg a \wedge b \wedge c) \vee (\neg a \wedge \neg b \wedge \neg c)$.

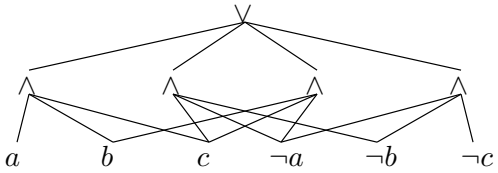


Figure 4. An element of the MODS fragment.

For the sake of completeness, we also consider the dual language of MODS, consisting of CNF formulae given by their *canonical implicates*:

- The language CI is the subset of CNF containing formulae $\Sigma = \gamma_1 \wedge \ldots \wedge \gamma_k$ such that for each clause γ_i $(i \in 1 \ldots k)$ and for each variable $x \in Var(\Sigma)$, γ_i contains x or $\neg x$.

Obviously enough, the negation of any MODS formula can be turned in linear time into a CI formula, and the converse also holds.

Now, it is well-known that eliminating a single quantification within an OBDD$_<$ formula can be achieved in time quadratic in the input size (an OBDD$_<$ formula equivalent to $\exists x.\Sigma$ (resp. $\forall x.\Sigma$) is computed as $\Sigma_{x \leftarrow 0} \vee \Sigma_{x \leftarrow 1}$, (resp. $\Sigma_{x \leftarrow 0} \wedge \Sigma_{x \leftarrow 1}$), see e.g. [39]). Since the size of the resulting formula may be quadratic in the size of the input Σ, there is no guarantee that such an elimination process leads to a formula of size polynomial in the input size when iterated so as to eliminate more than a preset number of variables. Hence, there is no guarantee that the time needed by such an elimination algorithm will remain polynomial in the input size. Actually, the next proposition shows that whatever the approach to solving VAL(QOBDD$_<$), a polytime algorithm is very unlikely:

Proposition 3. VAL(QDNNF), VAL(Qd-DNNF), VAL(QFBDD) *and* VAL($\mathtt{QOBDD}_<$) *are* PSPACE*-complete.*

Proof 1. *The membership directly comes from the fact that* VAL(QCNF) *is in* PSPACE*, the fact that the circuit language associated to* $\mathtt{PROP}_{PS}$ *includes* $\mathtt{NNF}_{PS}$ *as a proper subset, the fact that every circuit (encoding a boolean function over* $\{x_1, \ldots, x_n\}$*) can be mapped in polynomial time to a* CNF *formula over an extended set of variables, whilst equivalent to the circuit whenever the new variables are forgotten (i.e., existentially quantified) (see e.g. [43]), and the fact that* PSPACE *is closed under polynomial reduction.*

As to hardness, since the following inclusions hold (cf. Figure 5 in the appendix)

$$\mathtt{OBDD}_< \subset \mathtt{FBDD} \subset \mathtt{d\text{-}DNNF} \subset \mathtt{DNNF}$$

it is sufficient to prove that the VAL($\mathtt{QOBDD}_<$) *problem is* PSPACE*-hard. The proof is by reduction from* VAL(QDNF)*. The main step is to show that every* DNF *formula* $\phi = \gamma_1 \vee \ldots \vee \gamma_n$ *can be associated in polynomial time to an equivalent QBF of the form* $\exists X.\psi$ *where* $X \cap Var(\phi) = \emptyset$ *and* ψ *is from* $\mathtt{OBDD}_<$ *(whatever* $<$ *over* $Var(\phi)$*). First, let us note* $obdd(\gamma_i)$ *the* $\mathtt{OBDD}_<$ *formula equivalent to the term* γ_i *(*$i \in 1 \ldots n$*); clearly enough, every* $obdd(\gamma_i)$ *can be computed in time polynomial in* $|\gamma_i|$*. Let* $Var(\phi) = \{y_1, \ldots, y_m\}$ *and let* $X = \{x_1, \ldots, x_{n-1}\}$ *be a set of new variables; let* $\psi = \psi^1$*, where the formulae* ψ^i *(*$i \in 1 \ldots n$*) are defined by:*

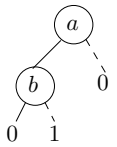
- $\psi^n = obdd(\gamma_n)$*, and*
- $\psi^i = (obdd(\gamma_i) \wedge x_i) \vee (\psi^{i+1} \wedge \neg x_i)$*, for* $i = 1, \ldots, n-1$.

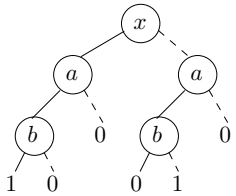
From such definitions, ψ *– which can be read as an* $\mathtt{OBDD}_<$ *formula where the new ordering* $<$ *is the extension of the previous ordering* $y_1 < \ldots < y_m$ *such that* $x_1 < \ldots < x_{n-1} < y_1 < \ldots < y_m$ *– can be computed in time polynomial in the size of* ϕ*. Now, since for every QBF* α*,* β *and every variable* x*, we have that* $\exists x.(\alpha \vee \beta) \equiv (\exists x.\alpha) \vee (\exists x.\beta)$ *(from point 6 in Proposition 1), and* $\exists x.(\alpha \wedge x) \equiv \exists x.(\alpha \wedge \neg x) \equiv \alpha$ *whenever* $x \notin Var(\alpha)$ *(from point 10 in Proposition 1), it immediately follows that* $\phi \equiv \exists X.\psi$*. Finally, the substitution metatheorem for QBF shows that for any prefix* P*, the QBF* $P.\phi$ *is equivalent to the QBF* $P\ \exists X.\psi$*, and this completes the proof.* □

The following example illustrates the proof above:

Example 9. *Let us consider the following* DNF *formula* $\phi = (\gamma_1 \vee \gamma_2)$ *where* $\gamma_1 = (a \wedge b)$ *and* $\gamma_2 = (a \wedge \neg b)$.

We have :

$\psi^2 \equiv$ 

So $\psi \equiv \psi^1 \equiv$ 

and $\phi \equiv \exists x.\psi$. ■

Based on the reduction given in the proof above, one can also show that the VAL($\texttt{QOBDD}_<$) problem spans all the polynomial hierarchy when restrictions are put on the prefix of the input: if no alternations of quantifiers occur, the problem reduces to the satisfiability problem or to the validity problem, and both of them are in P for $\texttt{OBDD}_<$ formulae; if the prefix is of the form $\forall S_1\ \exists S_2$, the problem is Π_1^p-complete (= coNP-complete); if the prefix is of the form $\exists S_1\ \forall S_2\ \exists S_3$, the problem is Σ_2^p-complete, and so on. Since the negation of an $\texttt{OBDD}_<$ formula can be computed as an $\texttt{OBDD}_<$ formula in constant time, we also obtain that if the prefix is of the form $\exists S_1\ \forall S_2$, the problem is Σ_1^p-complete (= NP-complete), if the prefix is of the form $\forall S_1\ \exists S_2\ \forall S_3$, the problem is Π_2^p-complete, and so on.

Adding the smoothness requirement to VAL(d-DNNF) does not help to reduce the complexity; indeed, every d-DNNF formula can be smoothed in polynomial time:

Corollary 1. VAL(Qsd-DNNF) *is* PSPACE*-complete.*

Proof 2. *Direct from the fact that the* VAL(Qd-DNNF) *is* PSPACE*-hard and that a* d-DNNF *can be turned into an equivalent* sd-DNNF *formula in polynomial time (see Lemma A.1 in [13]).* □

Contrastingly, VAL($\texttt{QOBDD}_<$) is tractable when the corresponding prefixes are compatible with the total, strict ordering $<$ associated with the $\texttt{OBDD}_<$ fragment:

Definition 7 (compatibility)**.** *Let* $\Sigma = QS_1 \ldots QS_n.\phi$ *be a prenex, polite, closed QBF where each* Q *stands for a quantifier and* $\{S_1, \ldots, S_n\}$ *is a partition of* $Var(\phi)$ *which does not contain the empty set. The prefix* $QS_1 \ldots QS_n$ *of* Σ *is said to be* compatible *with a total, strict ordering* $<$ *over* $Var(\phi)$ *if and only if for each* $x, y \in Var(\phi)$ *s.t.* $x < y$*, if* $x \in S_i$ *and* $y \in S_j$ *then* $j \geq i$*.*

Proposition 4. *The restriction of the* VAL($\texttt{QOBDD}_<$) *problem for instances with a prefix compatible with* $<$ *is in* P*.*

Proof 3. *The proof is constructive and is given by the polytime algorithm* SolveObdd *below. Such an algorithm consists in eliminating the quantifications (from the innermost to the outermost) into the input formula.*

Let x *be the greatest variable w.r.t.* $<$*.*

Let us first assume that x *is existentially quantified. By construction, an interpretation* I *is a model (resp. a counter-model) of an* $\texttt{OBDD}_<$ *formula* ϕ *if and only if the unique path from the root of* ϕ *that is compatible with* I *leads to the sink* 1 *(resp.* 0*); such a path corresponds to an implicant of* ϕ *(resp. its negation).*

Let N *be any decision node labeled by* x *in* ϕ*. Given the ordering assumption about* x*, the only possible children of* N *in* ϕ *are the sink nodes (actually, one of them is the* 1 *sink and the other one is the* 0 *sink if the ordered binary decision diagram* ϕ *is reduced, which can be assumed w.l.o.g. since such a reduction can be achieved in polynomial time).*

Since every model of $\exists x.\phi$ *coincides with a model of* ϕ *except possibly on* x*, it is sufficient to remove every node* N *labeled by* x *and to re-direct its incoming edges to the* 1 *sink to obtain an* $\texttt{OBDD}_<$ *formula equivalent to* $\exists x.\phi$*; the resulting formula is not necessarily reduced,*

but it can be reduced in polynomial time if it is not the case. The overall process can be easily achieved in time polynomial in the size of ϕ.

Now, if the greatest variable x w.r.t. $<$ is universally quantified, we take advantage of the equivalence $\forall x.\phi \equiv \neg(\exists x.\neg\phi)$ (point 3 in Proposition 1) and the fact that an $\mathtt{OBDD}_<$ formula equivalent to the negation of an $\mathtt{OBDD}_<$ formula can be obtained in constant time (just switch the labels of the two sink nodes). This completes the proof. □

Algorithm 1: Polytime algorithm for the restriction of VAL($\mathtt{QOBDD}_<$) to instances with a compatible prefix

procedure SolveObdd
Data: A formula $Q_1x_1 \ldots Q_nx_n.\phi$ from $\mathtt{QOBDD}_<$ where $\phi \in \mathtt{OBDD}_<$ is reduced and $x_1 < \cdots < x_n$
Result: 1 if $Q_1x_1 \ldots Q_nx_n.\phi$ is valid, 0 otherwise
begin
  for i **from** n **to** *1* **do**
    if $Q_i = \exists$ **then**
      foreach *node* N *labeled by* x_i **do**
        $N \leftarrow 1$;
    else
      foreach *node* N *labeled by* x_i **do**
        $N \leftarrow 0$;
    Reduce(ϕ) ;
  return ϕ ;
end

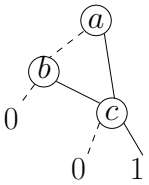
A Reduce procedure is described in [39], and a more efficient one, running in time linear in the input size, is given in [44]. As a consequence, SolveObdd runs in time $\mathcal{O}(n \times |\phi|)$.

The following example illustrates a run of SolveObdd:

Example 10. *Let $<$ be the ordering over $\{a, b, c\}$ such that $a < b < c$. Let us consider*

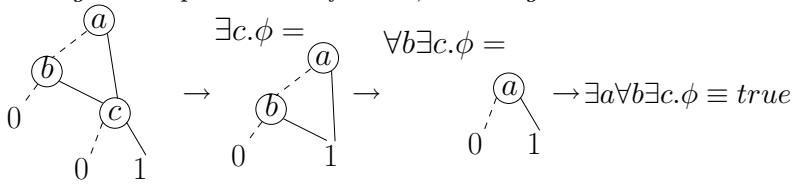
$$\Sigma = \exists a\ \forall b\ \exists c.\phi$$

$$\textit{with } \phi = (a \vee b) \wedge c$$

The $\mathtt{OBDD}_<$ formula associated to ϕ is: 

.

The algorithm proceeds as follows, allowing us to conclude that Σ is valid.



■

Compatibility proves to be a key for deciding the validity problem for QBF when restricted to other fragments. Thus, we have also considered the two propositional fragments $\texttt{ODNF}_<$ and $\texttt{OCNF}_<$ which are respectively a subset of DNF and of CNF. Let $<$ be a total, strict ordering over the variables from $PS = \{x_1, \ldots, x_n\}$ s.t. $x_1 < \ldots < x_n$. We define:

- The language $\texttt{ODNF}_<$ is the set of all DNF formulae $\Sigma = \gamma_1 \vee \ldots \vee \gamma_k$ where each satisfiable term γ_i $(i \in 1 \ldots k)$ of Σ is such that for every $j \in 1 \ldots n$, if x_j or $\neg x_j$ occurs in γ_i then for every l such that $1 \leq l < j$, x_l or $\neg x_l$ occurs in γ_i.
- The language $\texttt{OCNF}_<$ is the set of all CNF formulae $\Sigma = \gamma_1 \wedge \ldots \wedge \gamma_k$ where each non tautologous clause γ_i $(i \in 1 \ldots k)$ of Σ is such that for every $j \in 1 \ldots n$, if x_j or $\neg x_j$ occurs in γ_i then for every l such that $1 \leq l < j$, x_l or $\neg x_l$ occurs in γ_i.

Example 11. *Let $PS = \{a, b, c\}$ and $<$ such that $a < b < c$. $\neg a \vee (a \wedge b)$ is a formula from $\texttt{ODNF}_<$. $(a \vee b) \wedge (\neg a \vee b \vee c)$ is a formula from $\texttt{OCNF}_<$.* ■

Clearly enough, $\texttt{OCNF}_<$ (resp. $\texttt{QOCNF}_<$) is the dual fragment of $\texttt{ODNF}_<$ (resp. $\texttt{QODNF}_<$); especially the negation of any $\texttt{ODNF}_<$ (resp. $\texttt{QODNF}_<$) formula can be computed in linear time as a $\texttt{OCNF}_<$ (resp. $\texttt{QOCNF}_<$) formula, and the converse also holds. Furthermore, both $\texttt{OCNF}_<$ and $\texttt{ODNF}_<$ are complete propositional fragments since they include respectively the complete fragments MODS and CI.

Proposition 5. *The restrictions of* VAL($\texttt{QODNF}_<$) *and* VAL($\texttt{QOCNF}_<$) *to instances with a prefix compatible with $<$ are in* P.

Proof 4. *Since the negation of a $\texttt{QOCNF}_<$ formula can be computed in linear time as a $\texttt{QODNF}_<$ formula, it is sufficient to prove that* VAL($\texttt{QODNF}_<$) *is in* P *when restricted to instances with a prefix compatible with $<$.*

Again, the proof is constructive and is given by the polytime algorithm Solve$\texttt{ODNF}_<$ *below. This algorithm consists in eliminating quantifications as an internal law in the $\texttt{ODNF}_<$ fragment. In this algorithm, every $\texttt{ODNF}_<$ formula ϕ is considered w.l.o.g. as a set of terms and every term as a set of literals. After the first step of the algorithm, every term in ϕ is satisfiable (removing unsatisfiable terms in ϕ can be easily achieved in polynomial time). The resulting $\texttt{ODNF}_<$ formula is either the empty set (i.e., the empty disjunction equivalent to $false$ ($\{\} \equiv false$)) or a set containing the empty conjunction, equivalent to $true$ ($\{\{\}\} \equiv true$).*

Let us explain how quantifications can be eliminated and first consider the case of existential quantifiers; we take advantage of two equivalences: for every pair of QBF α and β and every variable $x \in PS$, we have $\exists x.(\alpha \vee \beta) \equiv (\exists x.\alpha) \vee (\exists x.\beta)$ (point 6 in Proposition 1), and $\exists x.\gamma$ is equivalent to the term $\gamma \setminus \{x, \neg x\}$ obtained by removing x and $\neg x$ from γ whenever γ is a satisfiable term (viewed as the set of its literals) (which is a direct consequence of the definition of conditioning and point 2 in Proposition 1). Let x be the last variable of $Var(\Sigma)$ w.r.t. $<$. Clearly enough, if γ is a canonical term over $X \cup \{x\}$ (with $x \notin X$), then $\gamma \setminus \{x, \neg x\}$ is a canonical term over X. Hence removing every occurrence of x and $\neg x$ in such an $\texttt{ODNF}_<$ formula ϕ leads to an $\texttt{ODNF}_<$ formula equivalent to $\exists x.\phi$.

Let us now focus on the case of universal quantifiers. Let ϕ be an $\texttt{ODNF}_<$ formula, viewed as a set of satisfiable terms over $X \cup \{x\}$, where $x \notin X$ and x is the last variable of $Var(\Sigma)$

w.r.t. $<$. *Each term of* ϕ *containing* x *or* $\neg x$ *is canonical over* $X \cup \{x\}$. *Now, the terms* γ *of* ϕ *can be partitioned into three sets (disjunctively interpreted):* W, S *and* S'. W *is the set of all terms* γ *s.t.* $x \notin Var(\gamma)$. W *can be viewed as an* $\mathtt{ODNF}_<$ *formula over* X. S *is the set of all terms* γ *of* ϕ *containing* x *or* $\neg x$ *s.t.* $switch(\gamma, x)$ *belongs to* ϕ *as well, where* $switch(\gamma, x)$ *is the canonical term over* $X \cup \{x\}$ *which coincides with* γ *for every variable of* X *but contains* $\neg x$ *whenever* γ *contains the literal* x, *while* $switch(\gamma, x)$ *contains* x *whenever* γ *contains* $\neg x$. *It is obvious that* γ *belongs to* S *if and only if* $switch(\gamma, x)$ *belongs to* S. *Finally,* S' *is the set of remaining terms from* ϕ *(i.e. every term* $\gamma \in S'$ *contains* x *or* $\neg x$ *but does not belong to* S*). By construction, we have* $\forall x.\phi \equiv \forall x.(W \vee S \vee S') \equiv W \vee \forall x.(S \vee S')$ *(from point 9 from Proposition 1). Let us now observe that* $S = \{\gamma_1, \ldots, \gamma_k\}$ *(disjunctively interpreted) is independent from* x *[45] in the sense that it is equivalent to the disjunction* $\psi = (\gamma_1 \setminus \{x, \neg x\}) \vee \ldots \vee (\gamma_k \setminus \{x, \neg x\})$ *which is an* $\mathtt{ODNF}_<$ *formula over* X *as well (for every* γ_i *(*$i \in 1 \ldots k$*), we have* $\gamma_i \vee switch(\gamma_i, x) \equiv \gamma_i \setminus \{x, \neg x\}$*). Clearly enough,* ψ *can be computed in time polynomial in the size of* ϕ. *Since* ψ *is independent from* x, *we get* $\forall x.\phi \equiv W \vee \psi \vee (\forall x.S')$. *Finally, it remains to show that* $\forall x.S'$ *is unsatisfiable (so that* $\forall x.\phi \equiv W \vee \psi$*). Let* γ *be any canonical term over* X *viewed as an interpretation over* X*; by definition,* γ *satisfies* $\forall x.S'$ *if and only if both the extension of it assigning* x *to* 0 *and the extension of it assigning* x *to* 1 *are models of* S'*; but this is impossible due to the definition of* S' *(if one of such extensions belongs to* S'*, it cannot be the case that the second one belongs to* S' *as well since it is obtained by switching* x *in the first one). Accordingly,* $(\forall x.S')$ *is unsatisfiable, and this concludes the proof.* □

Algorithm 2: Polytime algorithm for the restriction of VAL($\mathtt{QODNF}_<$) to instances with a compatible prefix

procedure SolveODNF$_<$
Data: A formula $Q_1x_1 \ldots Q_nx_n.\phi$ where $\phi \in \mathtt{ODNF}_<$ and $x_1 < \ldots < x_n$
Result: 1 if $Q_1x_1 \ldots Q_nx_n.\phi$ is valid, 0 otherwise
begin
  Remove unsatisfiable terms from ϕ ;
  for i **from** n **to** *1* **do**
    if $Q_i = \forall$ **then**
      $\phi' \leftarrow \{\gamma \in \phi \mid switch(\gamma, x_i) \in \phi$ or $x \notin Var(\gamma)\}$;
    else
      $\phi' \leftarrow \phi$;
    $\phi \leftarrow \{\gamma \setminus \{x_i, \neg x_i\} \mid \gamma \in \phi'\}$;
  if $\phi = \{\}$ **then**
    return 0 ;
  else
    return 1 ;
end

SolveODNF$_<$ runs in time $\mathcal{O}(n \times |\phi|^2)$. This algorithm could be easily refined, returning 0 as soon as the empty set has been obtained.

Let us illustrate how SolveODNF$_<$ works on two simple examples (where $a < b$):

Example 12.

1. $\forall a \exists b.\{\{\neg a\},\{a,\neg b\}\} \rightarrow \forall a.\{\{\neg a\},\{a\}\} \rightarrow \{\{\}\} \equiv true.$
2. $\exists a \forall b.\{\{\neg a, b\},\{a, b\}\} \rightarrow \exists a.\{\} \rightarrow \{\} \equiv false.$

■

As an immediate consequence, one gets the tractability of val(QMODS) and val(QCI):

Corollary 2. val(QMODS) *and* val(QCI) *are in* P.

Proof 5. *Direct from the fact that every formula from* QMODS *(resp.* QCI*) is a formula from* QODNF$_<$ *(resp.* QOCNF$_<$*) with a prefix compatible with* $<$*, whatever the strict, total ordering* $<$*.* □

Let us now turn to two additional, important propositional fragments in AI: the prime implicates one and the (dual) prime implicants one (see e.g., [46] for a survey of their applications in abduction, assumption-based reasoning, closed world reasoning and other AI areas). Formally, a *prime implicate* of $\phi \in PROP_{PS}$ is a clause δ s.t. $\phi \models \delta$ and for every clause δ' s.t. $\phi \models \delta'$ and $\delta' \models \delta$, we have $\delta \equiv \delta'$.

Definition 8 (prime implicates formulae)**.** *A prime implicates formula (or Blake formula) from* PROP$_{PS}$ *is a* CNF *formula* Σ *where every prime implicate of* Σ *(up to logical equivalence) appears as a conjunct.* PI *is the language of all prime implicates formulae (a proper subset of* CNF*).*

Example 13. *The following is a prime implicates formula:*

$$(a \vee b) \wedge (\neg b \vee c \vee d) \wedge (a \vee c \vee d).$$

■

The set of prime implicates formulae is a tractable fragment for SAT since (1) a CNF formula Σ is a prime implicates one if and only if no clause of it is properly entailed by another clause of Σ and every resolvent from two clauses from Σ is entailed by a clause of Σ (this shows that the problem of deciding whether a propositional formula is a prime implicates one can be decided in polynomial time), and (2) a prime implicates formula Σ is satisfiable if and only if it does not reduce to the empty clause.

Unfortunately, PI is not a tractable fragment for val(QPROP$_{PS}$) (under the usual assumptions of complexity theory):

Proposition 6. val(QPI) *is* PSPACE*-complete.*

Proof 6. *Membership comes directly from the fact that* val(QCNF) *is in* PSPACE *and every* PI *formula also is a* CNF *formula.*

As to hardness, let us give a polytime reduction from val(QCNF) *to* val(QPI)*. Let* ϕ *be a* CNF *formula over* $\{x_1, \ldots, x_n\}$*, viewed as the set* S *of its clauses. We take advantage of*

the following property, which results directly from the correctness of resolution-based prime implicates algorithms (like Tison's one [47]): a set S of clauses contains all its prime implicates if and only if whenever two clauses from S have a resolvent δ, there exists a clause $\epsilon \in S$ s.t. $\epsilon \models \delta$.

Let us assume that S is totally ordered w.r.t. an arbitrary (but fixed) ordering $<$. Let δ_i and δ_j be two clauses from S with $i < j$; when δ_i and δ_j have a resolvent, replace δ_i by $\delta_i \vee y_{i,j}$ and δ_j by $\delta_j \vee \neg y_{i,j}$; doing it in a systematic way for every ordered pair of clauses from S leads to generate in polynomial time a CNF *formula ψ over an extended vocabulary $\{x_1, \ldots, x_n\} \cup Y$ where $\mathcal{O}(n^2)$ new variables $y_{i,j}$ are introduced. By construction, every binary resolvent from clauses of ψ is tautologous, hence implied by any clause of ψ. As a consequence, ψ contains all its prime implicates, and a prime implicates formula θ equivalent to ψ can be computed in time polynomial in $|\psi|$, just by removing every clause of ψ which is properly implied by another clause from ψ.*

Now, for every pair of QBFs α and β and every variable $x \in PS$, we have $\forall x.(\alpha \wedge \beta) \equiv (\forall x.\alpha) \wedge (\forall x.\beta)$ (point 5 in Proposition 1); furthermore, for every non tautologous clause δ (viewed as the set of its literals) and every variable $x \in PS$, $\forall x.\delta$ is equivalent to the clause $\delta \setminus \{x, \neg x\}$ (this is a direct consequence of the definition of conditioning and point 1 in Proposition 1). As a consequence, we have $\phi \equiv \forall Y.\theta$. Finally, the substitution metatheorem for QBFs shows that $QS_1 \ldots QS_k.\phi$ is equivalent to $QS_1 \ldots QS_k \forall Y.\theta$, and this concludes the proof. □

The following example illustrates the reduction given in the above proof:

Example 14. *Let $\phi = (a \vee b) \wedge (\neg b \vee c \vee d) \wedge (\neg c \vee d)$ be a* CNF *formula. We associate in polynomial time ϕ to the following* PI *formula $\theta = (a \vee b \vee y_{1,2}) \wedge (\neg b \vee c \vee d \vee \neg y_{1,2} \vee y_{2,3}) \wedge (\neg c \vee d \vee \neg y_{2,3})$. We have $\phi \equiv \forall y_{1,2}, y_{2,3}.\theta$.* ■

Based on the reduction given in the proof above, one can also show that VAL(QPI) hits every level from the polynomial hierarchy when restrictions are put on the prefix: if no alternations of quantifiers occur, the problem is in P, if the prefix is of the form $\exists S_1\ \forall S_2$, the problem is NP-complete, if the prefix is of the form $\forall S_1\ \exists S_2\ \forall S_3$, the problem is Π_2^p-complete, and so on.

Now, what's about the restriction of VAL(QPI) for the instances for which the rightmost quantification of the prefix is existential? Contrariwise to $\text{OBDD}_<$ formulae in the general case, the negation of a PI formula cannot be computed in polynomial time (and even in *polynomial space*) as a PI formula, hence the same argument cannot be used again. Nevertheless, it is easy to show that a rightmost existential quantification does not lead to a complexity shift:

Proposition 7. *The restriction of* VAL(QPI) *to instances with prefixes of the form $\forall S_1\ \exists S_2$ is in* P.

Proof 7. *The proof is based on the fact that it is possible to eliminate in polynomial time existential quantifications as an internal law in the* PI *fragment. To be more precise, let $PI(\phi)$ be the set of prime implicates of a propositional formula ϕ (only one representative per equivalence class is kept); let us consider the following lemma (a direct consequence of Proposition 55 in [46]):*

Lemma 1. *Let ϕ be a formula from $\mathtt{PROP}_{PS}$ and let Y be a set of variables from PS. We have $PI(\exists Y.\phi) = \{\delta \in PI(\phi) \mid Var(\delta) \cap Y = \emptyset\}$.*

This lemma shows how a CNF *formula ψ over X which is equivalent to $\exists Y.\phi$ can be derived in time polynomial in $|\phi|$.*

Then we exploit the two following properties (already at work in the proof of Proposition 6): for every pair of QBF α and β and every variable $x \in PS$, we have $\forall x.(\alpha \wedge \beta) \equiv (\forall x.\alpha) \wedge (\forall x.\beta)$ and for every non tautologous clause δ (viewed as the set of its literals) and every variable $x \in PS$, $\forall x.\delta$ is equivalent to the clause $\delta \setminus \{x, \neg x\}$. A direct consequence of them is that $\forall X.\psi$ is valid if and only if ψ contains only tautologous clauses, which can be easily checked in time polynomial in $|\psi|$. □

The following example illustrates the lemma given in the above proof:

Example 15. *A* PI *formula equivalent to $\exists a.((a \vee b) \wedge (\neg b \vee c \vee d) \wedge (a \vee c \vee d))$ is $\neg b \vee c \vee d$.* ■

As a corollary to the previous proposition, we obtain that if the prefix of the input is of the form $\exists S_1\ \forall S_2\ \exists S_3$, the VAL(QPI) problem is NP-complete, if the prefix of the input is of the form $\forall S_1\ \exists S_2\ \forall S_3\ \exists S_4$, the problem is Π_2^p-complete, and so on.

The following dual class also is interesting. Let $\phi \in \mathtt{PROP}_{PS}$. A *prime implicant* of ϕ is a term γ s.t. $\gamma \models \phi$ and for every term γ' s.t. $\gamma' \models \phi$ and $\gamma \models \gamma'$, we have $\gamma \equiv \gamma'$.

Definition 9 (prime implicants formulae)**.** *A prime implicants formula from $\mathtt{PROP}_{PS}$ is a* DNF *formula Σ where every prime implicant of Σ (up to logical equivalence) appears as a disjunct.* IP *is the language of all prime implicants formulae (a proper subset of* DNF*).*

Prime implicants formulae are duals of prime implicates formulae: every prime implicant of a formula ϕ is (up to logical equivalence) the negation of a prime implicates of $\neg\phi$ (see e.g. Proposition 8 in [46]). As a consequence, QIP and QPI are also dual classes. Taking advantage of duality, we also obtain that:

Proposition 8. VAL(QIP) *is* PSPACE*-complete.*

Proof 8. *Immediate from Proposition 6 given the fact that* PSPACE *is closed under complementation and the fact that the negation of a* QPI *formula can be turned in linear time into a* QIP *formula.* □

Proposition 9. *The restriction of* VAL(QIP) *to instances with prefixes of the form $\exists S_1\ \forall S_2$ is in* P*.*

Proof 9. *This a direct consequence of Proposition 7 (by duality).* □

Exploiting duality, it is easy to show that the classes Σ_i^p and Π_i^p of the polynomial hierarchy that were not "hit" by restrictions of VAL(QPI) are "hit" by restrictions of VAL(QIP): if no alternations of quantifiers occur, the problem VAL(QIP) is in P, if the prefix of the input is of the form $\forall S_1\ \exists S_2$ or $\forall S_1\ \exists S_2\ \forall S_3$, the problem is coNP-complete, if the prefix of the input is of the form $\exists S_1\ \forall S_2\ \exists S_3$ or $\exists S_1\ \forall S_2\ \exists S_3\ \forall S_4$, the problem is Σ_2^p-complete, and so on.

4. Some Experimental Results

From the practical side, an important question is to determine how existing solvers for VAL($\texttt{QPROP}_{PS}$) behave on instances from tractable fragments. Especially, when solvers do not solve "easily" instances from tractable fragments, finding out the reason for it can prove a major step in the improvement of existing solvers. In order to answer this question, we performed some experiments on instances from QCI and $\texttt{QOCNF}_<$ under the compatibility assumption, which are tractable subsets of QCNF, the class of QBF almost all existing solvers deal with.

4.1 Experimental settings

We generated $\texttt{QOCNF}_<$ benchmarks with compatible prefixes in the following way. Given a number of variables n, each integer in $\{1, \ldots, n\}$ is identified with a variable. We took $<$ as the natural ordering on integers. In order to generate a clause of the matrix of a $\texttt{QOCNF}_<$ formula, the size s of the clause is first chosen at random in $\{1, \ldots, n\}$ under a uniform distribution, then the sign of each of the s variables (from 1 to s) of the clause is chosen at random under a uniform distribution.

For our experiments, we set n to 100, 200, 300, 400, 500, 1000 variables, which produce huge benchmarks, compared to the benchmarks commonly used for evaluating state-of-the-art solvers. The number of clauses was arbitrarily fixed to four times the number of variables ($m = 4 \times n$). The prefix is generated so as to ensure the compatibility with $<$. The number of quantifier alternations was fixed to 12: 11 sets of the same size (n div 11) and a last one with the remaining (n mod 11) variables. Note that this number of alternations is also quite high compared to the usual standards in QBF benchmarks.

We used three QBF solvers for those experiments. Our own Java-based QBF solver, called OpenQBF, which participated to the three QBF evaluations and reported medium strength results, while being among the few solvers not declared incorrect during the last two QBF evaluations. We also used Semprop release 010604 and Qube-Rel 1.3, two publicly available QBF solvers. The solvers ran on a PIV 3GHz with 1.5GB of RAM under Linux. The Java solver ran on the latest Sun Java VM (1.5.0_06) using default settings.

4.2 Results

Table 1 summarizes the behaviour of the three solvers on QCI and $\texttt{QOCNF}_<$ instances. Note that the running times are expressed in seconds, for solving 100 instances with the same parameters. The *size* column reports the total space taken by the generated benchmarks (for a total greater than 3 GB for those experiments!). The *#true* column reports the number of positive instances of QBF out of 100. One can note that all the instances from QCI were positive. This can be explained by the fact that each clause has roughly half of its literals existentially quantified (since by construction roughly half of the variables are existentially quantified and that the clauses contain all variables).

Note that all solvers perform better on $\texttt{QOCNF}_<$ benchmarks than on QCI benchmarks. This is probably due to the difference in the size of the benchmarks: QCI benchmarks are twice as big as $\texttt{QOCNF}_<$ benchmarks. Note also that it is usually easier to solve negative instances than positive ones: most of the instances in $\texttt{QOCNF}_<$ are negative ones.

Table 1. Running times of three solvers on instances from two polynomial classes of QBF. Cumulative CPU time in seconds for solving 100 instances for a given number of variables.

	CI					$\mathtt{OCNF}_<$ (compatible prefix)				
#vars	size	Own	Semp.	Qube	#true	size	Own	Semp.	Qube	#true
100	14	36	10	2	100	8	28	2	1	11
200	62	71	71	7	100	30	44	12	4	12
300	144	124	240	16	100	69	73	38	8	13
400	260	209	554	28	100	125	104	94	14	19
500	411	316	1062	43	100	198	145	165	21	10
1000	1680	1217	8638	174	100	824	532	1601	82	13

The three solvers were able to easily solve all benchmarks (a mean of 86 seconds per benchmark in the worst case, even if the running times ranges over one order of magnitude in that case). Note that those benchmarks are really huge compared to usual QBF benchmarks which explains why a usually robust solver such as Semprop does not scale well.

Those experiments show that the current solvers do not have problems for solving instances from QCI and $\mathtt{QOCNF}_<$. This is in contrast with our previous experiments on some other tractable (but incomplete) fragments for the validity problem (quantified Horn CNF and quantified renamable Horn CNF) [48] that were confirmed during the latest QBF evaluation [29].

5. Compilability

It is interesting to note the connection between the problem of finding out tractable matrix-based restrictions of VAL($\mathtt{QPROP}_{PS}$) and the compilability issue for VAL($\mathtt{QPROP}_{PS}$) when matrices are fixed while prefixes may vary. A basic issue is the compilability to P of VAL($\mathtt{QPROP}_{PS}$), i.e., the membership to the compilability class compP of the compilation problem COMP-VAL($\mathtt{QPROP}_{PS}$) associated to QBF where each instance is divided into two parts – the fixed one is the matrix and the variable one is the prefix [49, 50].

Definition 10 (COMP-VAL($\mathtt{QPROP}_{PS}$))**.** COMP-VAL($\mathtt{QPROP}_{PS}$) *is the language of pairs* $\{\langle \Sigma, P\rangle \mid P.\Sigma$ *is a closed, polite, prenex formula from* $\mathtt{QPROP}_{PS}$ *which is valid*$\}$.

Definition 11 (compP [50])**.** *A language of pairs* L *belongs to* compP *if and only if there exists a polysize function* f *and a language of pairs* $L' \in$ P *such that* $\langle x, y\rangle \in L$ *if and only if* $\langle f(x), y\rangle \in L'$.

Actually, the practical significance of such a compilability issue comes from the fact that, instead of considering the matrix as fixed, one can more generally consider the case when it can be computed in polynomial time from more basic inputs. Because many target classes $\mathcal{C}$ for knowledge compilation enable polytime conditioning, polytime bounded conjunction and polytime bounded disjunction [13], many inference problems can be encoded (and solved) as QBFs whose matrices in $\mathcal{C}$ can be computed in polynomial time from a compiled formula $\Sigma \in \mathcal{C}$ (the knowledge base) and a clausal query γ.

The membership of comp-val($\mathtt{QPROP}_{PS}$) to compP can be expressed by the following question: can we find a complete propositional fragment $\mathcal{C}$ for which there is a polynomial $p(.)$ s.t. every propositional formula α has an equivalent $\beta \in \mathcal{C}$ satisfying $|\beta| \leq p(|\alpha|)$ and there is a polytime algorithm for val($\mathtt{Q}\mathcal{C}$)? Clearly, $\mathcal{C} = \mathtt{MODS}$ (or $\mathtt{CI}$) is not a satisfying answer since it is not the case that every propositional formula has polynomially many models (or canonical implicates). Actually, it seems that no such fragment $\mathcal{C}$ exists:

Proposition 10. comp-val($\mathtt{QPROP}_{PS}$) *is not in* compP *unless* NP $\subseteq$ P/poly. *The conclusion holds under the restriction when at most one alternation of quantifiers occurs in the prefix of the input.*[5.]

Proof 10. *Let us first give a brief refresher about non-uniform complexity classes:*

Definition 12 (Advice-taking Turing machine)**.** *An advice-taking Turing machine is a Turing machine that has associated with it a special "advice oracle" A, which can be any function (not necessarily a recursive one). On input s, a special "advice tape" is automatically loaded with $A(|s|)$ and from then on the computation proceeds as normal, based on the two inputs, s and $A(|s|)$.*

Definition 13 (Polynomial advice)**.** *An advice-taking Turing machine uses polynomial advice if its advice oracle A satisfies $|A(n)| \leq p(n)$ for some fixed polynomial p and all non-negative integers n.*

Definition 14 ($\mathcal{C}$/poly)**.** *If $\mathcal{C}$ is a class of languages defined in terms of resource-bounded Turing machines, then $\mathcal{C}$/poly is the class of languages defined by Turing machines with the same resource bounds but augmented by polynomial advice.*

We show that the NP*-complete* 3-cnf-sat *problem can be solved in (deterministic) polynomial time using a Turing machine with polynomial advice if* comp-val($\mathtt{QPROP}_{PS}$) *belongs to* compP*. For each integer n, let Φ_n be the* CNF *formula (viewed as a set of clauses) containing all the clauses of the form $y_i \vee \delta_i$ where δ_i is a clause with at most 3 literals built up from the set X_n of variables $x_1, \ldots, x_n$ and each y_i is a new symbol (one for each clause δ_i). Clearly enough, Φ_n (up to logical equivalence) depends only on n. Furthermore, the number p_n of clauses in Φ_n is s.t. $p_n \in \mathcal{O}(n^3)$, hence the size of Φ_n is polynomial in n.*

Let ϕ_n be any CNF *formula over X_n. By construction, each clause δ_i from ϕ_n corresponds to a unique clause $y_i \vee \delta_i$ from Φ_n. Let ψ_n be the complement of $\{y_i \vee \delta_i \mid \delta_i \in \phi_n\}$ in Φ_n. One can easily compute in polynomial time from ϕ_n the set Y_{ϕ_n} of variables y_i s.t. the corresponding clauses δ_i belong to ϕ_n. Now, we know that for every pair of QBFs α and β and every variable $y \in PS$, we have $\forall y.(\alpha \wedge \beta) \equiv (\forall y.\alpha) \wedge (\forall y.\beta)$ (point 5 in Proposition 1); for every clause δ (viewed as the set of its literals) and every variable $y \in PS$, if δ does not contain both y and $\neg y$, then $\forall y.\delta$ is equivalent to the clause $\delta \setminus \{y, \neg y\}$ (as a direct consequence of the definition of conditioning and point 1 in Proposition 1); and if the variable y does not occur in the formula α, we have $\forall y.\alpha \equiv \alpha$ (an easy consequence of point 4 in Proposition 1); From those three properties, we get immediately that $\forall Y_{\phi_n}.\Phi_n \equiv \phi_n \wedge \psi_n$.*

5. Contrastingly, the restriction of comp-val($\mathtt{QPROP}_{PS}$) to instances where no alternation occurs obviously is in compP, just because there are only two possible prefixes for each matrix in this case (one is composed of existential quantifications only, and the other one of universal quantifications only).

Since every clause $y_i \vee \delta_i$ from ψ_n contains the literal y_i which is pure in $\phi_n \wedge \psi_n$ (i.e., its negation does not occur), it follows that $\phi_n \wedge \psi_n$ is satisfiable if and only if ϕ_n is satisfiable. Hence, ϕ_n is satisfiable if and only if $\forall Y_{\phi_n}.\Phi_n$ is satisfiable if and only if $\exists X_n\ \forall Y_{\phi_n}.\Phi_n$ is valid. Finally, let us assume that there exists a complete propositional fragment $\mathcal{C}$ for which VAL($\mathtt{Q}\mathcal{C}$) *is tractable and for which there is a polynomial $p(.)$ s.t. every propositional formula α has an equivalent $\beta \in \mathcal{C}$ satisfying $\beta \leq p(|\alpha|)$. Then* 3-CNF-SAT *can be solved in polynomial time by a deterministic Turing machine with advice as follows: for each input ϕ_n, the advice for n first gives a polyspace formula $\Phi_{n,\mathcal{C}}$ from $\mathcal{C}$ which is equivalent to Φ_n and then the satisfiability of ϕ_n is decided in polynomial time as the validity of $\exists X_n\ \forall Y_{\phi_n}.\Phi_{n,\mathcal{C}}$.* □

Note that NP $\subseteq$ P/poly would imply the polynomial hierarchy to collapse at the second level [51], which is considered very unlikely.

6. Conclusion

In this paper, we have presented new tractability and new intractability results for the validity problems for QBFs when the matrices of the inputs belong to a target class for knowledge compilation. In the light of our study, the complexity landscape for VAL($\mathtt{QPROP}_{PS}$) can be completed as reported in Table 2.

Fragment $\mathcal{C}$	**Complexity of** VAL($\mathtt{Q}\mathcal{C}$)
$\mathtt{PROP}_{PS}$ (general case)	PSPACE-c
CNF	PSPACE-c
DNF	PSPACE-c
d-DNNF	PSPACE-c
sd-DNNF	PSPACE-c
DNNF	PSPACE-c
FBDD	PSPACE-c
$\mathtt{OBDD}_{<}$	PSPACE-c
PI	PSPACE-c
IP	PSPACE-c
$\mathtt{OBDD}_{<}$ (compatible prefix)	$\in$ P
$\mathtt{ODNF}_{<}$ (compatible prefix)	$\in$ P
$\mathtt{OCNF}_{<}$ (compatible prefix)	$\in$ P
MODS	$\in$ P
CI	$\in$ P

Table 2. Complexity results for VAL($\mathtt{QPROP}_{PS}$).

In [13], the authors have investigated the spatial efficiency of several complete propositional fragments, including many of those considered in this paper. A given fragment $\mathcal{C}_1$ is considered at least as concise as a second fragment $\mathcal{C}_2$ whenever there exists a polynomial $p(.)$ s.t. for every formula $\alpha \in \mathcal{C}_2$, there exists an equivalent formula $\beta \in \mathcal{C}_1$ s.t. $|\beta| \leq p(|\alpha|)$.

Our results show that VAL($\mathtt{QPROP}_{PS}$) is difficult even when limited to instances with matrices from fragments which are not efficient from the spatial point of view (i.e., the $\mathtt{OBDD}_<$ one, the PI fragment and the IP fragment). Tractability is achieved without restrictions only for the MODS fragment and the CI fragment which are among the least efficient ones (as to spatial efficiency) (see [13]). Under the compatibility assumption, tractability is also achieved for the more concise $\mathtt{OBDD}_<$, $\mathtt{ODNF}_<$ and $\mathtt{OCNF}_<$ fragments; those fragments appear as the best candidates among the classes considered in this paper which enable tractable QBF queries; however, the choice of the ordering $<$ has a major impact on the size of the formulae (see [39] for the $\mathtt{OBDD}_<$ case).

This work calls for several perspectives. One of them consists in further extending the complexity landscape of VAL($\mathtt{QPROP}_{PS}$), focusing on other complete or incomplete fragments. In particular, it would be interesting to determine how the notion of K-Boolean model from [52], in the case when K is a class of boolean functions which can be encoded in polynomial space, could be exploited to give rise to new restrictions of VAL($\mathtt{QPROP}_{PS}$) which are computationally easier (under the usual assumptions of complexity theory).

While the focus has been laid on matrix-based restrictions of VAL($\mathtt{QPROP}_{PS}$) in this paper, it might seem quite natural at a first glance to consider prefix-based restrictions as well. However, from a theoretical point of view, considering restrictions on the prefix alone (i.e., without further requirements on the matrix like we did it before when considering the $\mathtt{OBDD}_<$ fragment, the $\mathtt{OCNF}_<$ fragment, the $\mathtt{ODNF}_<$ fragment, the prime implicates fragment and the prime implicants fragment) does not look so much interesting. While limiting the number of quantifications alternations in the prefix leads immediately the complexity of VAL($\mathtt{QPROP}_{PS}$) to decrease from PSPACE to a given level in the polynomial hierarchy, it does not lead to tractability (under the standard assumptions of complexity theory); indeed, in the limit case, no quantification alternations occur in the prefix so that the VAL($\mathtt{QPROP}_{PS}$) problem reduces either to SAT or to UNSAT (depending of the nature of the quantification of all variables), and none of these problems is likely to belong to P provided that no assumption on the matrix is made. Of course, this does not mean that the way quantifications are processed has no practical impact on the efficiency of solvers for VAL($\mathtt{QPROP}_{PS}$). For instance, QBFs Σ of the form $\forall\{x_1,\ldots,x_{n-1}\}\exists\{x_n\}.\phi$ where ϕ is a CNF formula can be solved in quadratic time (compute a CNF representation of $\exists\{x_n\}.\phi$ using resolution to forget x_n within ϕ, remove tautologous clauses, then shorten every resulting clause by removing from it every literal built up from $\{x_1,\ldots,x_{n-1}\}$; Σ is valid if and only if the resulting CNF formula does not contain the empty clause). However, solvers for VAL($\mathtt{QPROP}_{PS}$) where quantifications are processed from the outermost to the innermost may require an exponential amount of time to solve some Σ of this kind. A deeper investigation of the interaction between prefix-based restrictions and matrix-based ones is a perspective for further research.

Acknowledgments

Many thanks to the Région Nord / Pas-de-Calais through the IRCICA Consortium and the COCOA project, and to the IUT de Lens for their support. Special thanks to the anonymous reviewers for their very constructive remarks and suggestions (especially for suggesting the tractability of VAL($\mathtt{QODNF}_<$) and VAL($\mathtt{QOCNF}_<$) under the compatibility assumption).

Appendix

Figure 5 depicts the inclusion graph of the propositional fragments considered in this paper.

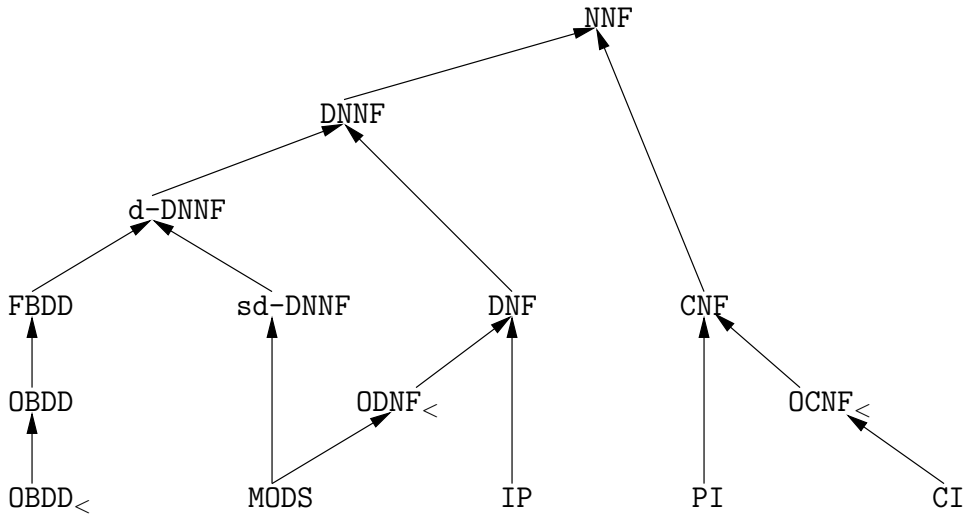


Figure 5. Inclusion graph of propositional fragments. An edge $L_1 \rightarrow L_2$ means that L_1 is a proper subset of L_2.

References

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

[2] A. Del Val. Tractable databases: how to make propositional unit resolution complete through compilation. In *KR'94*, pages 551–561, 1994.

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

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

[5] S. Coste-Marquis and P. Marquis. Knowledge compilation for circumscription and closed world reasoning. *Journal of Logic and Computation*, **11**(4):579–607, 2001.

[6] S. Coste-Marquis and P. Marquis. On stratified belief base compilation. *Annals of Mathematics and Artificial Intelligence*, **42**(4):399–442, 2004.

[7] A. Darwiche and P. Marquis. Compiling propositional weighted bases. *Artificial Intelligence*, **157**(1–2):81–113, 2004.

[8] S. Coste-Marquis and P. Marquis. Characterizing consistency-based diagnoses. In *AI & Math'98*, 1998.
http://rutcor.rutgers.edu/~amai/aimath98.

[9] A. Darwiche. Compiling devices into decomposable negation normal form. In *IJCAI'99*, pages 284–289, 1999.

[10] A. Cimmati, E. Giunchiglia, F. Giunchiglia, and P. Traverso. Planning via model checking: a decision procedure for AR. In *ECP'97*, pages 130–142, 1997.

[11] H. Geffner. Planning graphs and knowledge compilation. In *KR'04*, pages 662–672, 2004.

[12] H. Palacios, B. Bonet, A. Darwiche, and H. Geffner. Pruning conformant plans by counting models on compiled d-DNNF representations. In *ICAPS'05*, pages 141–150, 2005.

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

[14] U. Egly, T. Eiter, H. Tompits, and S. Woltran. Solving advanced reasoning tasks using Quantified Boolean Formulas. In *AAAI'00*, pages 417–422, 2000.

[15] H. Fargier, J. Lang, and P. Marquis. Propositional logic and one-stage decision making. In *KR'00*, pages 445–456, 2000.

[16] J. Rintanen. Constructing conditional plans by a theorem-prover. *Journal of Artificial Intelligence Research*, **10**:323–352, 1999.

[17] G. Pan, U. Sattler, and M.Y. Vardi. BDD-based decision procedures for K. In *CADE'02*, pages 16–30, 2002.

[18] Ph. Besnard, T. Schaub, H. Tompits, and S. Woltran. *Inconsistency tolerance*, volume **3300** of *LNCS State-of-the-Art Survey*, chapter Representing paraconsistent reasoning via quantified propositional logic, pages 84–118. Springer-Verlag, 2005.

[19] M. Cadoli, A. Giovanardi, and M. Schaerf. An algorithm to evaluate Quantified Boolean Formulae. In *AAAI'98*, pages 262–267, 1998.

[20] J. Rintanen. Improvements to the Evaluation of Quantified Boolean Formulae. In *IJCAI'99*, pages 1192–1197, 1999.

[21] R. Feldmann, B. Monien, and S. Schamberger. A distributed algorithm to evaluate quantified boolean formulas. In *AAAI'00*, pages 285–290, 2004.

[22] E. Giunchiglia, M. Narizzano, and A. Tacchella. Backjumping for Quantified Boolean Logic satisfiability. In *IJCAI'01*, pages 275–281, 2001.

[23] R. Letz. Lemma and model caching in decision procedures for Quantified Boolean Formulas. In *Tableaux'02*, pages 160–175, 2002.

[24] L. Zhang and S. Malik. Towards a symmetric treatment of satisfaction and conflicts in quantified boolean formula evaluation. In *CP'02*, pages 200–215, 2002.

[25] G. Pan and M.Y. Vardi. Symbolic decision procedures for QBF. In *CP'04*, pages 453–467, 2004.

[26] G. Audemard and L. Saïs. SAT based BDD solver for Quantified Boolean Formulas. In *ICTAI'04*, pages 82–89, 2004.

[27] D. Le Berre, L. Simon, and A. Tacchella. Challenges in the QBF arena: the SAT'03 evaluation of QBF solvers. In *SAT'03*, volume **2919** of *LNCS*, pages 468–485, 2003.

[28] D. Le Berre, M. Narizzano, L. Simon, and A. Tacchella. The second QBF solvers evaluation. In *SAT'04*, volume **3542** of *LNCS*, pages 376–392, 2004.

[29] M. Narizzano, L. Pulina, and A. Tacchella. The third QBF solvers comparative evaluation. *Journal on Satisfiability, Boolean Modeling and Computation*, **2**:145–164, 2006.

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

[31] N. Creignou, S. Khanna, and M. Sudan. Complexity classification of boolean constraint satisfaction problems. In *SIAM Monographs on Discrete Mathematics and Applications*, volume **7**. Society for Industrial and Applied Mathematics, 2001.

[32] B. Aspvall, M. Plass, and R. Tarjan. A linear-time algorithm for testing the truth of certain quantified boolean formulas. *Information Processing Letters*, **8**:121–123, 1979. Erratum: Information Processing Letters 14(4): 195 (1982).

[33] H. Kleine Büning, M. Karpinski, and A. Flögel. Resolution for quantified boolean formulas. *Information and Computation*, **117**(1):12–18, 1995.

[34] I. P. Gent and A. G. D. Rowley. Solving 2-CNF Quantified Boolean Formulae using variable assignment and propagation. In *QBF workshop at SAT'02*, pages 17–25, 2002.

[35] F. Börner, A. Bulatov, A. Krokhin, and P. Jeavons. Quantified constraints: algorithms and complexity. In *CSL'03*, volume **2803** of *LNCS*, pages 58–70, 2003.

[36] H. Chen. Collapsibility and consistency in quantified constraint satisfaction. In *AAAI'04*, pages 155–160, 2004.

[37] H. Chen. Quantified constraint satisfaction, maximal constraint languages, and symmetric polymorphisms. In *STACS'05*, pages 315–326, 2005.

[38] Ch. H. Papadimitriou. *Computational complexity*. Addison–Wesley, 1994.

[39] R. E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Trans. on Computers*, C-**35**(8):677–692, 1986.

[40] U. Egly, M. Seidl, H. Tompits, S. Woltran, and M. Zolda. Comparing different prenexing strategies for Quantified Boolean Formulas. In *SAT'03*, volume **2919** of *LNCS*, pages 214–228, 2003.

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

[42] J. Gergov and C. Meinel. Efficient analysis and manipulation of OBDDs can be extended to FBDDs. *IEEE Trans. on Computers*, **43**(10):1197–1209, 1994.

[43] X. Zhao and H. Kleine Büning. Model-equivalent reductions. In *SAT'05*, volume **3569** of *LNCS*, pages 355–370, 2005.

[44] D. Sieling and I. Wegener. Reduction of OBDDs in linear time. *Information Processing Letters*, **48**(3):139–144, 1993.

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

[46] P. Marquis. *Consequence finding algorithms*, volume **5** of *Handbook on Defeasible Reasoning and Uncertainty Management Systems*, chapter 2, pages 41–145. Kluwer Academic Publisher, 2000.

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

[48] S. Coste-Marquis, D. Le Berre, and F. Letombe. A branching heuristics for quantified renamable Horn formulas. In *SAT'05*, volume **3569** of *LNCS*, pages 393–399, 2005.

[49] P. Liberatore. Monotonic reductions, representative equivalence, and compilation of intractable problems. *Journal of the ACM*, **48**(6):1091–1125, 2001.

[50] M. Cadoli, F.M. Donini, P. Liberatore, and M. Schaerf. Preprocessing of intractable problems. *Information and Computation*, **176**(2):89–120, 2002.

[51] R. M. Karp and R. J. Lipton. Some connections between non-uniform and uniform complexity classes. In *STOC'80*, pages 302–309, 1980.

[52] H. Kleine Büning, K. Subramani, and X. Zhao. On boolean models for quantified boolean Horn formulas. In *SAT'03*, volume **2919** of *LNCS*, pages 93–104, 2003.