Function problems for Quantified Boolean Formulas

S. Coste-Marquis, H. Fargier, J. Lang, D. Le Berre and P. Marquis

July 18, 2003

Abstract

The practical use of Quantified Boolean Formulas (QBFs) often calls for more than solving the evaluation problem QBF. For this reason we investigate the corresponding function problem FQBF. We define expected outputs of FQBF instances as solution policies. We focus on the representation of policies, considering QBFs of the form $\forall X\ \exists Y\ \Phi$. Because the explicit representation of policies for such QBFs can be of exponential size, descriptions as compact as possible must be looked for. To address this issue, two approaches based on the decomposition and the compilation of Φ are presented.

1 Introduction

1.1 Motivations

A *Quantified Boolean Formula* (QBF) consists of a classical propositional formula together with an ordered partition of its variables, corresponding to quantifier alternations; for instance, if Φ is a formula built up from the set of propositional variables $\{a, b, c, d\}$, then $\exists\{a\}\ \forall\{b, d\}\ \exists\{c\}\ (((a \wedge \neg c) \rightarrow (b \wedge d)) \wedge (b \rightarrow a))$ is a QBF. Any QBF evaluates to true or false; it evaluates to true if and only if the corresponding statement where quantifiers on variables bear actually on the *truth values* of these variables, holds, and in that case the QBF is said to be *positive*. Thus, the latter QBF is positive since there exists an instantiation of a such that for any instantiation of b and d there exists an instantiation of c such that $(((a \wedge \neg c) \rightarrow (b \wedge d)) \wedge (b \rightarrow a))$ is satisfied. QBF is the decision problem which consists in determining whether a given QBF is positive.

Solving the decision problem QBF has become for a few years an important research area in AI. Several explanations for this can be advanced, including the fact that many AI problems whose complexity is located in PSPACE can be expressed and then efficiently solved by QBF solvers [7]. Accordingly, many such solvers have been developed for the past few years (see mainly [2, 13, 14, 9, 11, 16]).

Interestingly, QBFs can also be used to represent some planning problems under incomplete knowledge and feedback and some sequential two-player games with complete information. For instance, $\exists\{a\}\forall\{b\}\exists\{c\}\forall\{d\}\Phi$ represents a game with two players $J_\forall$ and $J_\exists$, playing alternatively by assigning a propositional variable: $J_\exists$ starts by assigning a value to a, then $J_\forall$ assigns a value to b, etc. The goal of $J_\exists$ is to have

Φ satisfied at the end of the game: thus, $\exists\{a\}\forall\{b\}\exists\{c\}\forall\{d\}\Phi$ is a positive instance of QBF if and only if there exists a winning strategy for $J_\exists$. When the game is understood as a game against nature (or equivalently a planning problem with nondeterministic actions or exogenous events), instantiations of existentially (resp. universally) quantified variables correspond to plays by the agent (resp. by nature), and winning strategies are policies (or conditional plans).

Clearly enough, when QBFs are used to represent such problems, what is expected is often more than simply solving QBF. Indeed, solving the decision problem only enables telling whether there exists a winning strategy or a valid plan; in practice, one would also like to determine such a plan, or, at least, an approximation of it. Therefore, the aim becomes solving the *function problem* associated with QBFs, denoted by FQBF. We are not aware of general definitions for this function problem in the literature. Our first objective therefore consists in defining formally what an output of FQBF is. We call it a *solution policy*. In order to deal with the case where no solution policy exists, we show how to approximate them using partial policies. Then we focus on the representation of policies, considering QBFs of the form $\forall X\ \exists Y\ \Phi$. Because the explicit representation of policies for such QBFs can be of exponential size, descriptions as compact as possible must be looked for. To address this issue, two approaches based on the decomposition and the compilation of Φ are presented.

1.2 Notations and background

$PROP_{PS}$ denotes the propositional language built up from a finite set PS of symbols, the usual connectives and the boolean constants $\top$, $\bot$ in the standard way.

$\vec{x}$ is an instantiation of variables from $X \subseteq PS$ (also referred to as an X-instantiation) and 2^X is the set of all possible X-instantiations. Thus, if $X = \{a, b, c\}$, $\vec{x} = (a, \neg b, c)$ is an X-instantiation. If X and Y are two disjoint subsets of $PROP_{PS}$, $(\vec{x}, \vec{y})$ is the concatenation of $\vec{x}$ and $\vec{y}$: in this instantiation, each variable of X (resp. Y) takes the value indicated by $\vec{x}$ (resp. $\vec{y}$).

For $\Phi \in PROP_{PS}$ and $\vec{x} \in 2^X$, we denote by $\Phi_{\vec{x}}$ the formula obtained by conditioning Φ by $\vec{x}$; this formula is obtained from Φ by replacing occurrences of each variable x from X by $\top$ (resp. $\bot$) if $x \in \vec{x}$ (resp. $\neg x \in \vec{x}$).

Definition 1 (Quantified Boolean Formula) *Let k be a positive integer and $q \in Q$. A* Quantified Boolean Formula (QBF) *is a (k+3)-uple $P = \langle k, q, X_k, ..., X_1, \Phi\rangle$ where $\{X_1, ..., X_k\}$ is a partition of the set $Var(\Phi)$ of propositional variables occurring in $\Phi \in PROP_{PS}$.*

In theory, there is no need to specify k, since it can be determined from the number of elements of P. However, we keep it for the sake of readibility. For similar reasons, we also denote QBFs $P = \langle k, q, X_k, ..., X_1, \Phi\rangle$ in the following way (as in the previous subsection):

$\exists X_k \forall X_{k-1} \dots \forall X_1 \Phi$ if $q = \exists$ and k is even
$\exists X_k \forall X_{k-1} \dots \exists X_1 \Phi$ if $q = \exists$ and k is odd
$\forall X_k \exists X_{k-1} \dots \exists X_1 \Phi$ if $q = \forall$ and k is even
$\forall X_k \exists X_{k-1} \dots \forall X_1 \Phi$ if $q = \forall$ and k is odd

QBF$_{k,q}$ is the set of all QBFs of rank k and first quantifier q. We now define formally the decision problem QBF through its positive instances:

Definition 2 (positive instance of QBF)
$P = \langle k, q, X_k, ..., X_1, \Phi\rangle$ is a positive *instance of* QBF *if and only if one of the following conditions is true:*

- *$k = 0$ and $\Phi = \top$;*
- *$k \geq 1$ and $q = \exists$ and there exists an X_k-instantiation $\vec{x_k} \in 2^{X_k}$ such that $\langle k-1, \forall, X_{k-1}, ..., X_1, \Phi_{\vec{x_k}}\rangle$ is a positive instance of* QBF$_{k-1,\forall}$ *;*
- *$k \geq 1$ and $q = \forall$ and for each X_k-instantiation $\vec{x_k} \in 2^{X_k}$, $\langle k-1, \exists, X_{k-1}, \ldots, X_1, \Phi_{\vec{x_k}}\rangle$ is a positive instance of* QBF$_{k-1,\exists}$.

In this definition, QBF$_{k,q}$ is the subproblem of QBF where only formulas from QBF$_{k,q}$ are considered.

2 The function problem FQBF

As evoked in the introduction, solving planning problems or games represented as QBFs may require more than just solving the decision problem QBF. Stepping back to a previous example, knowing that $\exists\{a\}\forall\{b\}\exists\{c\}\forall\{d\}\Phi$ is a positive instance of QBF just tells that there is a truth value for a for which whatever the truth value of b there is a truth value for c that makes Φ true, independently of the truth value of d; it does not tell at all which truth value must be given to a and which truth value must be given to c (depending of the truth value of b) so as to make Φ true. This requires more sophisticated outputs than boolean ones, and we call such outputs *policies*.

Intuitively, a policy is an application mapping each instantiation of a group of universally quantified variables into an instantiation of the group of existentially quantified variables immediately following it. Formally, we start with the notion of *total policy*:

Definition 3 (total policy) *The set $TP(k, q, X_k, ..., X_1)$ of* total policies *for QBFs from QBF$_{k,q}$ is defined inductively by:*

- *$TP(0, q) = \{\lambda\}$;*
- *$TP(k, \exists, X_k, ..., X_1) =$*
  $\{\vec{x_k}\ ; \pi_{k-1} \mid \pi_{k-1} \in TP(k-1, \forall, X_{k-1}, ..., X_1)\}$;
- *$TP(k, \forall, X_k, ..., X_1) =$*
  $2^{X_k} \rightarrow TP(k-1, \exists, X_{k-1}, ..., X_1)$[1].

λ represents the *empty policy*. The operator ";" represents the sequential composition of policies. $\pi; \lambda$ is typically abbreviated as π. One can check that:

- a policy of $TP(1, \exists, X_1)$ has the form $(\vec{x_1}; \lambda)$, i.e., $\vec{x_1}$ (an X_1-instantiation);
- $TP(1, \forall, X_1)$ is reduced to a unique policy: the constant function which maps any X_1-instantiation to λ. This policy is also represented by the constant λ_{X_1};

[1] $2^{X_k} \rightarrow TP(k-1, \exists, X_{k-1}, ..., X_1)$ denotes the set of all total functions from 2^{X_k} to $TP(k-1, \exists, X_{k-1}, ..., X_1)$.

- a policy of $TP(2, \exists, X_2, X_1)$ has the form $(\vec{x_2}\ ; \lambda_{X_1})$;
- a policy of $TP(2, \forall, X_2, X_1)$ is a total function from 2^{X_2} to 2^{X_1}.

Definition 4 (satisfaction by a policy) *The satisfaction of an instance $P = \langle k, q, X_k, ..., X_1, \Phi\rangle$ of QBF$_{k,q}$ by a policy π of $TP(k, q, X_k, ..., X_1)$ is defined inductively as follows; π satisfies P (denoted $\pi \models P$) if and only if one of these conditions is verified:*

- *$k = 0$ and $\pi = \lambda$, and $\Phi \equiv \top$;*
- *$k \geq 1$ and $q = \exists$ and $\pi = (\vec{x_k}; \pi')$ with $\pi' \models \langle k-1, \forall, X_{k-1}, ..., X_1, \Phi_{\vec{x_k}}\rangle$;*
- *$k \geq 1$ and $q = \forall$ and for all $\vec{x_k} \in 2^{X_k}$ we have $\pi(\vec{x_k}) \models \langle k-1, \exists, X_{k-1}, ..., X_1, \Phi_{\vec{x_k}}\rangle$.*

Example 1 *There is no policy satisfying $\langle 2, \forall, \{a, b\}, \{c\}, (a \vee b) \wedge (a \rightarrow c) \wedge (b \vee c)\rangle$.*

Example 2 *$\langle 3, \exists, \{a\}, \{b\}, \{c, d\}, (a \rightarrow (c \wedge d)) \wedge (b \leftrightarrow \neg c)\rangle$ is satisfied by $\pi = \neg a; \begin{bmatrix} (b), & \mapsto (\neg c, d) \\ (\neg b) & \mapsto (c, d) \end{bmatrix}$.*

It can be checked that $\pi = \vec{x_1}$ satisfies $P = \langle 1, \exists, X_1, \Phi\rangle$ if and only if $\vec{x_1} \models \Phi$ and that $\pi = \lambda_{X_1}$ satisfies $P = \langle 1, \forall, X_1, \Phi\rangle$ if and only if Φ is valid. More generally, we have the following result, which shows that total policies satisfying P are the right outputs of the function problem associated to P. They are called *solution policies* for P:

Proposition 1 *$P = \langle k, q, X_k, ..., X_1, \Phi\rangle$ is a positive instance of* QBF$_{k,q}$ *if and only if there exists a total policy $\pi \in TP(k, q, X_k, ..., X_1)$ such that $\pi \models P$*[2].

This result enables us to define formally the function problem FQBF$_{k,q}$ as follows:

Definition 5 (FQBF: function problem)
Let $P = \langle k, q, X_k, ..., X_1, \Phi\rangle$ be a QBF. Solving the function problem FQBF$_{k,q}$ *for P consists in finding a total policy π such that $\pi \models P$, if there exists any.*

3 Partial policies

In practice, asking for a solution policy is often too much demanding. Indeed, let us consider $P = \langle 2, \forall, \{a, b\}, \{c\}, (a \rightarrow c) \wedge (b \rightarrow \neg c)\rangle$. P does not have a solution policy because the instantiation (a, b) makes Φ unsatisfiable: thus, if nature plays (a, b), the agent cannot do anything leading to the satisfaction of Φ. On the other hand, if nature plays anything but (a, b) then the agent *can* do something satisfactory, namely, $(a, \neg b) \mapsto c$, $(\neg a, b) \mapsto \neg c$, $(\neg a, \neg b) \mapsto c$ (or $\neg c$). These policies are not defined for all possible instantiations of groups of universally quantified variables, hence their name *partial policies*. Here is a formal definition:

[2]For space reasons, proofs are typically omitted.

Definition 6 (partial policy) *The set $PP(k, q, X_k, ..., X_1)$ of* partial policies *for the QBF $P = \langle k, q, X_k, \ldots, X_1\rangle$ is defined inductively as follows:*

- $PP(1, \exists, X_1) = 2^{X_1} \cup \{\times\}$ *;*
- $PP(1, \forall, X_1) = 2^{X_1} \to \{\lambda, \times\}$ *;*
- $PP(k, \exists, X_k, ..., X_1) =$
  $\{\vec{x_k}; \pi_{k-1} \,|\pi_{k-1} \in PP(k-1, \forall, X_{k-1}, .., X_1) \cup \{\times\}\}$*;*
- $PP(k, \forall, X_k, ..., X_1) =$
  $2^{X_k} \to PP(k-1, \exists, X_{k-1}, ..., X_1)$.

$\times$ represents failure (i.e., it is not possible to find a solution policy for the considered QBF). Any partial policy from $PP(k-1, q, X_{k-1}, .., X_1)$ used to define a partial policy π of rank k along the definition above is called an internal policy *of π. It is a* universal internal policy *when $q = \forall$, and an* existential internal policy *otherwise.*

Definition 7 (sound policy) *A partial policy $\pi \in PP(k, q, X_k, ..., X_1)$ is* sound *for $P = \langle k, q, X_k, ..., X_1, \Phi\rangle$ if and only if one of these conditions is satisfied:*

1. *$q = \exists$ and $\pi = \times$;*
2. *$(k, q) = (1, \exists)$, $\pi = \vec{x_1}$ and $\vec{x_1} \models \Phi$;*
3. *$(k, q) = (1, \forall)$ and for any $\vec{x_1} \in 2^{X_1}$ we have either $\pi(\vec{x_1}) = \times$, or ($\pi(\vec{x_1}) = \lambda$ and $\vec{x_1} \models \Phi$) ;*
4. *$k > 1$, $q = \exists$, $\pi = \vec{x_k}; \pi_{k-1}$ and π_{k-1} is sound for $\langle k-1, \forall, X_{k-1}, ..., X_1, \Phi_{\vec{x_k}}\rangle$;*
5. *$k > 1$, $q = \forall$, and for any $\vec{x_k} \in 2^{X_k}$, $\pi(\vec{x_k})$ is sound for $\langle k-1, \exists, X_{k-1}, ..., X_1, \Phi_{\vec{x_k}}\rangle$.*

While there exist QBFs which do not have any solution policy, it is clear that all QBFs have a sound partial policy.

Example 3 *As Example 1 showed, there is no solution policy for $P = \langle 2, \forall, \{a, b\}, \{c\}, (a \vee b) \wedge (a \to c) \wedge (b \vee c)\rangle$. A sound policy for P is*

$$\pi = \begin{bmatrix} (a, b) & \mapsto c \\ (\neg a, b) & \mapsto c \\ (a, \neg b) & \mapsto c \\ (\neg a, \neg b) & \mapsto \times \end{bmatrix}.$$

Intuitively, the best policies among the sound ones are those built up from internal policies where $\times$ is used as less as possible. Formally:

Definition 8 (maximal sound policy) *Let π and π' two partial policies of $PP(q, k, X_k, ..., X_1)$. π is* at least as covering as *π', denoted by $\pi \sqsupseteq \pi'$, if and only if one of the following conditions is satisfied:*

- *$q = \exists$ and $\pi' = \times$;*
- *$q = \forall$, $k = 1$ and for all $\vec{x_1} \in 2^{X_1}$, we have either $\pi'(\vec{x_1}) = \times$ or $\pi(\vec{x_1}) = \lambda$;*

- $q = \exists$, $\pi = [\vec{x_k}; \pi_{k-1}]$, $\pi' = [\vec{x'_k}; \pi'_{k-1}]$, *and* $\pi_{k-1} \sqsupseteq \pi'_{k-1}$ *;*
- $q = \forall$, $k > 1$ *and for all* $\vec{x_k} \in 2^{X_k}$*, we have* $\pi(\vec{x_k}) \sqsupseteq \pi'(\vec{x_k})$*.*

$\sqsupseteq$ *is a partial preorder;* π *is a* maximal sound *policy for a QBF* P *if and only if* π *is sound for* P *and there is no sound policy* π' *for* P *such that* $\pi' \sqsupseteq \pi$ *and* $\pi \not\sqsupseteq \pi'$*.*

Example 4 *The QBF* $P = \langle 2, \forall, \{a, b\}, \{c\}, (a \vee b) \wedge (a \rightarrow c) \wedge (b \vee c)\rangle$ *has two maximal sound policies. The first one is reported in Example 3, the other one is identical to it, except that it maps* $(\neg a, b)$ *to* $\neg c$*.*

Clearly enough, every QBF P has a maximal sound policy; furthermore, if a solution policy for P exists, then solution policies and maximal sound policies coincide.

Definition 9 (SFQBF: second function problem)
Let $P = \langle k, q, X_k, ..., X_1, \Phi\rangle$ *be a QBF. Solving the second function problem* $\text{SFQBF}_{k,q}$ *for* P *consists in finding a maximal sound policy* π *for* P*.*

4 Policy representation

It is essential to make a distinction between the notion of policy π *per se* and the notion of *representation* σ of a policy. Indeed, policies may admit many different representations, and two representations of the same policy can easily have different sizes, and can be processed more or less efficiently (e.g. computing the image of an instantiation by a given universal internal policy can be more or less computationally demanding).

A *representation scheme* $\mathcal{S}$ for policies is a finite set of data structures representing policies. Associated with any representation scheme $\mathcal{S}$ is an interpretation function $I_{\mathcal{S}}$ such that for any $\sigma \in \mathcal{S}$, $\pi = I_{\mathcal{S}}(\sigma)$ is the policy represented by σ. The simplest representation scheme is the *explicit* one: the representation of a policy is the policy itself (so the corresponding interpretation function is identity). Accordingly, π also denotes the explicit representation of policy π. Within the explicit representation of a policy π, every universal internal policy π' is represented explicitly as a set of pairs (this is the representation we used in the examples reported in the previous sections). Clearly enough, a structured way of envisioning the explicit representation of a policy π of $PP(k, q, X_k, ..., X_1)$ is to consider a tree whose leaves are labelled by $\times$, λ or by instantiations of X_1, whose intermediate nodes are labelled by instantiations of groups of existentially quantified variables, and whose branches are labelled by instantiations of groups of universally quantified variables.

The next proposition makes precise the connection between the decision problem QBF and the function problem FQBF. It shows that explicit representations of total policies are *certificates* for QBF, i.e., data structures from which a polytime verification of the validity of positive instances is possible. To be more precise:

Proposition 2 *There is a polytime algorithm whose input is the explicit representation of a policy* $\pi \in TP(k, q, X_k, ..., X_1)$ *and a QBF* $P = \langle k, q, X_k, ..., X_1, \Phi\rangle$ *and which returns* 1 *if* π *is a solution policy for* P *and* 0 *otherwise.*

For every instance $P = \langle 1, \exists, X_1, \Phi \rangle$ of $\text{QBF}_{1,\exists}$ (i.e., every SAT instance), a solution policy π for P is represented explicitly by any model of Φ over X_1; obviously, such representations of policies are certificates for $\text{QBF}_{1,\exists}$.

Now, for every instance $P = \langle 1, \forall, X_1, \Phi \rangle$ of $\text{QBF}_{1,\forall}$, the solution policy π for P is represented explicitly by the set $\{(\vec{x_1}, \lambda) \mid \vec{x_1} \in 2^{X_1}\}$; again, this representation is a certificate for $\text{QBF}_{1,\forall}$. As evoked previously, the same policy π can be represented in an exponentially more succinct way as the constant λ_{X_1}; obviously, such an (non-explicit) representation of π is not a certificate for $\text{QBF}_{1,\forall}$, unless P = NP; furthermore, the existence of a certificate of polynomial size for $\text{QBF}_{1,\forall}$ would lead to NP = coNP, hence the polynomial hierarchy to collapse at the first level. This example clearly shows how different representations of the same policy may lead to different computational behaviours when the purpose is to use the policy.

5 The case of $\text{SFQBF}_{2,\forall}$

We now focus on the practical resolution of $\text{SFQBF}_{2,\forall}$. Why the choice of $k = 2$ and $q = \forall$? First, it is important, before investigating more complex $\text{SFQBF}_{k,q}$ problems, to focus first the problems of the first levels (which are already complex enough, as we will see). The case $k = 1$ has received an enormous attention; $\text{SFQBF}_{2,\exists}$ is not really new either, since it reduces to an abduction problem: indeed, it consists in finding an instantiation $\vec{x_1}$ such that $\Phi_{\vec{x_1}}$ is valid; this problem has been considered many times. Things are different with $\text{SFQBF}_{2,\forall}$, since (i) finding maximal sound policies becomes here relevant and (ii) the size of the representation of a policy becomes a crucial issue.

5.1 Polynomially compact and tractable schemes

In the case of $\text{SFQBF}_{2,\forall}$, a partial policy for $P = \langle 2, \forall, X, Y, \Phi \rangle$ is any mapping π from 2^X to $2^Y \cup \{\times\}$. Ideally, we are looking for representation schemes for maximal sound policies that are both polynomially compact and tractable:

Definition 10 (polynomially compact scheme) *A policy representation scheme $\mathcal{S}$ for maximal sound policies for QBF$_{2,\forall}$ is said to be* polynomially compact *if and only if there is a polysize function $R_{\mathcal{S}}$ that associates each $P = \langle 2, \forall, X, Y, \Phi \rangle \in$ QBF$_{2,\forall}$ to a representation $\sigma \in \mathcal{S}$ of a maximal sound policy π for P.*

Definition 11 (tractable scheme) *A policy representation scheme $\mathcal{S}$ for maximal sound policies for QBF$_{2,\forall}$ is said to be* tractable *if and only if there exists a polytime algorithm $D_{\mathcal{S}}$ such that for any $\sigma \in \mathcal{S}$, $D_{\mathcal{S}}$ computes $\pi(\vec{x}) = D(\sigma, \vec{x})$ for any $\vec{x} \in 2^X$, where $\pi = I_{\mathcal{S}}(\sigma)$.*

Clearly, the explicit representation scheme for maximal sound policies is not polynomially compact in the general case. For instance, there exist QBFs of QBF$_{2,\forall}$ for which any solution policy is injective, as shown by the following example (from [8]):

Example 5 $\forall\{x_1, \ldots, x_n\}\exists\{y_1, \ldots, y_n\} \bigwedge_{i=1}^{n}(x_i \leftrightarrow y_i)$

However, it is possible to encode the solution policies π for the set of QBFs of Example 5 (with n varying), using data structures σ of size polynomial in n and from which $\pi(\vec{x})$ can be computed in time polynomial in n. See for instance the policy description scheme *PD* given in the next subsection. This argues towards using implicit representation schemes for policies, but still, the existence of a polynomially compact and tractable representation scheme for maximal sound policies cannot be ensured:

Proposition 3 *If a polynomially compact and tractable representation scheme $\mathcal{S}$ for maximal sound policies for QBF$_{2,\forall}$ exists, then the polynomial hierarchy collapses at the second level.*

Proof: Suppose that there exists a representation scheme $\mathcal{S}$ such that there exists a polysize function $R_\mathcal{S}$ and a polytime algorithm $D_\mathcal{S}$ such that $R_\mathcal{S}$ maps each QBF $P = \langle 2, \forall, X, Y, \Phi \rangle$ to a tractable representation $\sigma = R_\mathcal{S}(P)$ of a maximal sound policy $\pi = I_\mathcal{S}(\sigma)$ and such that $D_\mathcal{S}$ computes $\pi(\vec{x}) = D(\sigma, \vec{x})$ for any $\vec{x} \in 2^X$.

Let us now consider the following nondeterministic algorithm for solving QBF$_{2,\forall}$:

```
Input:  a QBF $P = \langle 2, \forall, X, Y, \Phi \rangle$.
1.  guess $\sigma = R_\mathcal{S}(P)$;
2.  check that $\pi = I_\mathcal{S}(\sigma)$ is a solution policy for $P$.
```

Provided that $R_\mathcal{S}$ exists, guessing σ in step 1. only requires polynomial time (since its size must be polynomial in the input size).

Let us recall that P is a positive instance of QBF$_{2,\forall}$ if and only if it has a solution policy (Proposition 1), and that if a solution policy exists, then any maximal sound policy is a solution policy. Now, provided that $D_\mathcal{S}$ exists, when the input is P and σ has been guessed, the problem of determining whether $\pi = I_\mathcal{S}(\sigma)$ is *not* a solution policy for P is in NP: just guess $\vec{x} \in 2^X$ and check in polynomial time using $D_\mathcal{S}$ that $\pi(\vec{x}) = D_\mathcal{S}(\sigma, \vec{x}) = \times$. Accordingly, step 2. can be achieved using one call to an NP oracle.

Subsequently, the algorithm above shows that QBF$_{2,\forall}$ is in Σ_2^p, hence $\Sigma_2^p = \Pi_2^p$ and the polynomial hierarchy collapses at the second level. ■

Therefore we just want to look for representations of policies, which are *as concise as possible*, and especially more concise than the explicit representations, provided that they are tractable:

Definition 12 (tractable representation) *A representation σ of a policy π for a QBF $P = \langle 2, \forall, X, Y, \Phi \rangle \in$ QBF$_{2,\forall}$ is said to be* tractable *if and only if there exists an algorithm D_σ such that for any $\vec{x} \in 2^X$, D_σ computes $\pi(\vec{x}) = D(\vec{x})$ in time polynomial in $|\sigma| + |\vec{x}|$.*

5.2 The decomposition approach

It is based on two simple observations:

1. It is often needless looking for a specific Y-instantiation for each X-instantiation: some Y-instantiations may cover large sets of X-instantiations, which can be described in a compact way, for instance by a propositional formula.

2. It may be the case that some sets of variables from Y are more or less independent given X w.r.t. Φ and therefore that their assigned values can be computed

separately[3].

Definition 13 (subdecision) *An instantiation of* some *(not necessarily all) variables of Y (or equivalently, a satisfiable term γ_Y on Y) is called a* subdecision. 3^Y *is the set of all subdecisions. Any mapping $\pi : 2^X \rightarrow 3^Y$ assigning a subdecision to each X-instantiation is called a* subpolicy *for $\forall X \exists Y \Phi$. Similarly as for policies, we can also define* partial subpolicies *that assign a subdecision to a subset of 2^X. The* merging *of subdecisions is the commutative and associative internal operator on $3^Y \cup \{\times\}$ defined by:*

- $\gamma_Y.\lambda = \lambda.\gamma_Y = \gamma_Y$;
- $\gamma_Y.\times = \times.\gamma_Y = \times$;
- *if γ_Y, γ'_Y are two terms on Y, then* $\gamma_Y.\gamma'_Y = \begin{cases} \gamma_Y \wedge \gamma'_Y & \textit{if } \gamma_Y \wedge \gamma'_Y \ is\ satisfiable \\ \times & \textit{otherwise} \end{cases}$.

Here, the empty decision λ is assimilated to the empty term. The merging *of two subpolicies π_1, π_2 is defined by: $\forall \vec{x} \in 2^X$, $(\pi_1 \odot \pi_2)(\vec{x}) = \pi_1(\vec{x}).\pi_2(\vec{x})$.*

Definition 14 (policy description) *The* policy description scheme PD *is a representation scheme for maximal sound policies for QBF$_{2,\forall}$, defined inductively as follows:*

- *λ and $\times$ are in* PD*;*
- *any satisfiable term γ_Y on Y is in* PD*;*
- *if φ_X is a propositional formula built on X and σ_1, σ_2 are in* PD*, then* `if` φ_X `then` σ_1 `else` σ_2 *is in* PD[4]*;*
- *if σ_1 and σ_2 are in* PD*, then $\sigma_1 \odot \sigma_2$ is in* PD*.*

Now, the partial subpolicy $\pi = I_{PD}(\sigma)$ induced by a description $\sigma \in$ PD *is defined inductively as follows; for every $\vec{x} \in 2^X$:*

- *$I_{PD}(\lambda)(\vec{x}) = \lambda$ and $I_{PD}(\times)(\vec{x}) = \times$;*
- *$I_{PD}(\gamma_Y)(\vec{x}) = \gamma_Y$;*
- $I_{PD}(\texttt{if } \varphi_X \texttt{ then } \sigma_1 \texttt{ else } \sigma_2)(\vec{x}) =$ $\begin{cases} I_{PD}(\sigma_1)(\vec{x}) & \textit{if } \vec{x} \models \varphi_X \\ I_{PD}(\sigma_2)(\vec{x}) & \textit{if } \vec{x} \models \neg\varphi_X \end{cases}$
- *$I_{PD}(\sigma_1 \odot \sigma_2)(\vec{x}) = I_{PD}(\sigma_1(\vec{x})).I_{PD}(\sigma_2(\vec{x}))$.*

[3]As briefly evoked in [12] (Section 6), such independence properties can also prove helpful in the practical solving of instances of QBF.

[4]To simplify notations, (`if` φ `then` σ `else` $\times$) is abbreviated into `if` φ `then` σ and (`if` φ_1 `then` σ_1 `else...else if` φ_n `then` σ_n) into (`Case` φ_1: σ_1; ...; φ_n: σ_n `End`).

Example 6 *Let* $\sigma_1 = \texttt{if } x_1 \leftrightarrow x_2 \texttt{ then } y_1 \texttt{ else } \neg y_1, \sigma_2 = \texttt{if } x_1 \texttt{ then } \neg y_2$, *and* $\sigma = \sigma_1 \odot \sigma_2$. *The corresponding policies are given by*

	$I_{PD}(\sigma_1)$	$I_{PD}(\sigma_2)$	$I_{PD}(\sigma)$
(x_1, x_2)	y_1	$\neg y_2$	$(y_1, \neg y_2)$
$(x_1, \neg x_2)$	$\neg y_1$	$\neg y_2$	$(\neg y_1, \neg y_2)$
$(\neg x_1, x_2)$	$\neg y_1$	$\times$	$\times$
$(\neg x_1, \neg x_2)$	y_1	$\times$	$\times$

Proposition 4 PD *is a tractable representation scheme for maximal sound policies for QBF*$_{2,\forall}$.

Example 7 *A tractable representation in* PD *of the solution policy for the QBF* $\forall\{x_1, \ldots, x_n\}\exists\{y_1, \ldots, y_n\} \bigwedge_{i=1}^{n}(x$
$y_i)$ *considered in Example 5 is*

$$\sigma = \odot_{i=1}^{n}((\texttt{if } x_i \texttt{ then } y_i) \odot (\texttt{if } \neg x_i \texttt{ then } \neg y_i))$$

Proposition 5 *Let* $P = \langle 2, \forall, X, Y, \Phi\rangle$ *and let* $\{\varphi_1^X, \varphi_1^Y, \ldots, \varphi_p^X, \varphi_p^Y\}$ *be* $2p$ *formulas such that*

$$\Phi \equiv (\varphi_1^X \wedge \varphi_1^Y) \vee \ldots \vee (\varphi_p^X \wedge \varphi_p^Y).$$

Let $J = \{j \mid \varphi_j^Y$ *is satisfiable*$\} = \{j_1, \ldots, j_q\}$ *and for every* $j \in J$, *let* $\vec{y}_j \models \varphi_j^Y$. *Then the policy* π *represented by the description*

$$\sigma = \texttt{Case } \varphi_{j_1}^X\text{: } \vec{y}_{j_1}; \ldots; \varphi_{j_q}^X\text{: } \vec{y}_{j_q} \texttt{ End}$$

is a maximal sound policy for P.

The interest of Proposition 5 is that once Φ has been decomposed in such a way, the resolution of the instance of SFQBF given by $P = \forall X \exists Y \Phi$ comes down to solving p instances of SAT. Furthermore, it is always possible to find such a decomposition – just take all instantiations of X: $\Phi \equiv \bigvee_{\vec{x} \in 2^X}(\vec{x} \wedge \Phi_{\vec{x}})$.

Of course, such a decomposition is interesting only if it is not too large, i.e., if it leads to a reasonable number of SAT instances to solve. Let $N(\Phi)$ the minimal number of pairs of such a decomposition: the best case is $N(\Phi) = 1$ and the worst is $N(\Phi) = 2^{\min(|X|,|Y|)}$. Finding a good decomposition actually amounts to break the links between X and Y in Φ, the ideal case being when there are no links between them, i.e., when $\Phi \equiv \varphi_X \wedge \varphi_Y$ (or equivalently, X and Y are marginally conditionally independent with respect to Φ [3, 10]).

Furthermore, Proposition 5 immediately tells how to compute a maximal sound policy in polynomial time for $\forall X \exists Y \Phi$ when Φ is in DNF. Interestingly, the problem of computing of maximal sound policy (i.e., a solution policy when the $\forall X \exists Y \Phi$ is positive) is *easier* than the decision problem of deciding whether $\forall X \exists Y \Phi$ is positive (coNP-complete when Φ is a DNF formula).

The next decomposition result makes possible to compute subpolicies independently on disjoints subsets of Y, and then merge these subpolicies.

Proposition 6 *Let* $\{Y_1, Y_2\}$ *be a partition of* Y *such that* Y_1 *and* Y_2 *are conditionally independent given* X *with respect to* Φ, *which means that there exist two formulas* φ_{X,Y_1} *and* φ_{X,Y_2} *of respectively* $PROP_{X \cup Y_1}$ *and* $PROP_{X \cup Y_2}$ *such that*

$\Phi \equiv \varphi_{X,Y_1} \wedge \varphi_{X,Y_2}$. *Then π is a maximal sound policy for $\forall X \exists Y \Phi$ if and only if there exist two subpolicies π_1, π_2, which are maximal and sound for $\forall X \exists Y \varphi_{X,Y_1}$ and for $\forall X \exists Y \varphi_{X,Y_2}$ respectively, such that $\pi = \pi_1 \odot \pi_2$.*

Proposition 6 can be used efficiently to reduce an instance of SFQBF$_{2,\forall}$ into two (or several, when iterated) instances of SFQBF$_{2,\forall}$ with smaller sets Y. Ideally, Φ is already on the desired form (i.e., there exists a partition that works); however, in general this is not the case and we have then to find a candidate partition $\{Y_1, Y_2\}$ which is *almost* independent w.r.t. Φ given X, and then break the links between Y_1 and Y_2 through case-analysis on a set of variables from Y, which must be chosen as small as possible (for efficiency reasons). The good point is that we can take advantage of existing decomposition techniques to achieve that goal, especially those based on the notion of decomposition tree (see e.g. [4]).

5.3 The compilation approach

It consists in generating first a *compiled form* σ of Φ using any knowledge compilation algorithm.

Proposition 7 *Let $P = \forall X\ \exists Y\ \Phi$ be a QBF and let σ be a propositional formula equivalent to Φ and which belongs to a propositional fragment $\mathcal{F}$ enabling polytime clausal query answering, polytime conditioning and polytime model finding (see [6]). σ is a tractable representation of a maximal sound policy for P.*

Proof: Given an instantiation $\vec{x} \in 2^X$, the model of $\sigma_{\vec{x}}$ computed in polytime by the algorithm (whose existence is postulated above) is (if it exists) an instantiation $\vec{y} \in 2^Y$. Now, we can show that $\vec{y} \models \sigma_{\vec{x}}$ holds if and only if $(\exists Y \sigma)_{\vec{x}}$ is valid (indeed, a central property of conditioning is that an instantiation $\vec{x}$ is an implicant of a formula Ψ if and only if the conditioning $\Psi_{\vec{x}}$ is valid). Now, $(\exists Y \sigma)_{\vec{x}}$ is valid if and only if $\exists Y(\sigma_{\vec{x}})$ is valid (since $X \cap Y = \emptyset$). Lastly, remark that $\exists Y(\sigma_{\vec{x}})$ is valid if and only if $\sigma_{\vec{x}}$ is satisfiable. ■

Note that there is no policy representation scheme here. Actually, within this compilation-based approach, σ alone does not represent any policy for P but a specific maximal sound policy for P is fully characterized by the way a model of $\sigma_{\vec{x}}$ is computed for each $\vec{x}$.

Among the target fragments $\mathcal{F}$ of interest are all polynomial CNF classes for SAT problem, which are stable by conditioning (i.e., conditioning always leads to a CNF formula belonging to the class). Indeed, for every formula from such a class, polytime model enumeration is possible (see e.g. [6]). Among the acceptable classes are the Krom one, the Horn CNF one, and more generally the renamable Horn CNF one. Several other propositional fragments can be considered, including the DNF one, the OBDD one and more generally the DNNF one since each of them satisfies the three requirements imposed in Proposition 7.

Even if there is no guarantee that for every Φ, the corresponding σ is polysize (unless the polynomial hierarchy collapses at the second level), many experiments reported in [15, 1, 5] showed the practical interest of knowledge compilation techniques for clausal entailment; clearly, such a conclusion can be drawn as well when the purpose is the representation of tractable policies for QBFs from QBF$_{2,\forall}$.

6 Conclusion

In this paper we provided theoretical ground for resolution of function problems associated with QBFs, as well as some algorithmic techniques for solving (and representing solutions) of formulas $\forall X \; \exists \; \Phi$ from QBF$_{2,\forall}$.

A next step would consist in determining from the practical side the performances of several representation schemes for maximal sound policies. We plan to make some experiments to measure how the size of the representation of policies varies with various parameters (e.g., the numbers of clauses and of variables in a CNF of Φ, the ratio $\frac{|X|}{|X|+|Y|}$), and to compare it with the coverage of the corresponding policy (i.e., how many $\vec{x} \in 2^X$ are not mapped to $\times$).

References

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

[2] Marco Cadoli, Andrea Giovanardi, and Marco Schaerf. An algorithm to evaluate quantified boolean formulae. In *AAAI'98*, pages 262–267, 1998.

[3] A. Darwiche. A logical notion of conditional independance : properties and applications. *Artificial Intelligence*, 97(1–2):45–82, 1997.

[4] A. Darwiche. Decomposable negation normal form. *Journal of the Association for Computing Machinery*, 48(4):608–647, 2001.

[5] A. Darwiche. A compiler for deterministic decomposable negation normal form. In *AAAI'02*, pages 627–634, 2002.

[6] A. Darwiche and P. Marquis. A perspective on knowledge compilation. In *IJCAI'01*, pages 175–182, 2001.

[7] U. Egly, T. Eiter, H. Tompits, and S. Woltran. Solving advanced reasoning tasks using quantified boolean formulas. In *AAAI'00*, pages 417–422, 2000.

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

[9] Enrico Giunchiglia, Massimo Narizzano, and Armando Tacchella. Backjumping for quantified boolean logic satisfiability. In *IJCAI'01*, pages 275–281, 2001.

[10] J. Lang, P. Liberatore, and P. Marquis. Conditional independence in propositional logic. *Artificial Intelligence*, 141(1–2):75–121, 2002.

[11] Reinhold Letz. Advances in decision procedures for quantified boolean formulas. In *QBF Workshop at IJCAR'01*, pages 55–64, 2001.

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

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

[14] J. Rintanen. Partial implicit unfolding in the Davis-Putnam procedure for Quantified Boolean Formulae. In *QBF Workshop at IJCAR'01*, pages 84–93, 2001.

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

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