Representing Policies for Quantified Boolean Formulas

Sylvie Coste-Marquis
CRIL — Univ. d'Artois, Lens, France

Hélène Fargier and **Jérôme Lang**
IRIT — Univ. Paul Sabatier, Toulouse, France

Daniel Le Berre and **Pierre Marquis**
CRIL — Univ. d'Artois, Lens, France

Abstract

The practical use of Quantified Boolean Formulas (QBFs) often calls for more than solving the validity problem QBF. For this reason we investigate the corresponding function problems whose expected outputs are policies. QBFs which do not evaluate to true do not have any solution policy, but can be of interest nevertheless; for handling them, we introduce a notion of partial policy. 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.

Introduction

A Quantified Boolean Formula (QBF) consists of a classical propositional formula together with an ordered partition of its variables, corresponding to quantifier alternations, such as, for instance, $\exists\{a\}\ \forall\{b,d\}\ \exists\{c\}\ (a \wedge \neg c) \rightarrow (b \wedge d)$. 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 *valid* (as it is the case for the latter instance). QBF is the decision problem consisting in determining whether a given QBF is valid. 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 solved by QBF solvers (Egly *et al.* 2000). Accordingly, many such solvers have been developed for the past few years (see among others (Cadoli, Giovanardi, & Schaerf 1998; Rintanen 1999b; Feldmann, Monien, & Schamberger 2000; Giunchiglia, Narizzano, & Tacchella 2001; Letz 2002; Zhang & Malik 2002; Pan & Vardi 2004); see also (Littman 1999; Majercik & Littman 1999) for a probabilistic version of QBF.

Obviously, QBFs can be viewed as planning problems under incomplete knowledge and feedback as well as sequential two-player games with complete information. Clearly enough, when QBFs are used to represent such problems, what is expected is 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 (that we call a *solution policy*) or at least, an approximation of it. Therefore, the aim becomes solving the *function problem* associated with QBFs, denoted by FQBF.

While this function problem is nothing really new (it has been considered before in (Kleine-Büning, Subramani, & Zhao 2003; Liberatore 2005), as well as in (Chen 2004) in the close framework of quantified constraint satisfaction problems (QCSPs)), this paper investigates new issues. First, when no solution policy exists, we search for *partial policies* that solve the problem "as much as possible". Then, we introduce *representation schemes* of such policies. Lastly, we investigate the search for *compact policies*, focusing on 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 are looked for. This issue is addressed by two approaches, based respectively on the *decomposition* and the *compilation* of Φ.

Formal Preliminaries

In the rest of the paper, $PROP_{PS}$ 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 (respectively Y) takes the value indicated by $\vec{x}$ (respectively $\vec{y}$).

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 $\top$ (respectively $\bot$) if $x \in \vec{x}$ (respectively $\neg x \in \vec{x}$). For any sets E and F, $E \rightarrow F$ denotes the set of all total functions from E to F.

Let k be a positive integer and $q \in \{\exists, \forall\}$. A 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 of propositional variables occurring in $\Phi \in PROP_{PS}$. $P = \langle k, q, X_k, ..., X_1, \Phi\rangle$ is valid if and

only if one of the following three conditions is true:

1. $k = 0$ and $\Phi = \top$;
2. $k \geq 1$, $q = \exists$, 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 $\text{QBF}_{k-1,\forall}$;
3. $k \geq 1$, $q = \forall$, 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 $\text{QBF}_{k-1,\exists}$.

$\text{QBF}_{k,q}$ is the set of all QBFs of rank k and first quantifier q. $\text{QBF}_{k,q}$ is the subproblem of QBF where only formulas from $\text{QBF}_{k,q}$ are considered.

Policies

Intuitively, a policy is a function mapping instantiations of each group of universally quantified variables into instantiation of the group of existentially quantified variables immediately following it.

Definition 1 (total policy) *The set $TP(k, q, X_k, ..., X_1)$ of* total policies *for QBFs from $\text{QBF}_{k,q}$ is defined inductively by:*

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

For instance, 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. A policy of $TP(2, \forall, X_2, X_1)$ is a total function from 2^{X_2} to 2^{X_1}.

Definition 2 (satisfaction) *A total policy π of $TP(k, q, X_k, ..., X_1)$ satisfies $P = \langle k, q, X_k, ..., X_1, \Phi \rangle$, denoted by $\pi \models P$, if and only if one of these conditions holds:*

- $k = 0$ *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 $\langle 3, \exists, \{a\}, \{b\}, \{c, d\}, (a \rightarrow (c \wedge d)) \wedge (b \leftrightarrow \neg c) \rangle$ *is satisfied by* $\pi = \neg a; \begin{bmatrix} (b) & \mapsto (\neg c, d) \\ (\neg b) & \mapsto (c, d) \end{bmatrix}$.

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

Now, asking for a solution policy is often too much demanding. For instance, let us consider $P = \langle 2, \forall, \{a, b\}, \{c\}, (a \rightarrow c) \wedge (b \rightarrow \neg c) \rangle$: P is not valid 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 $\neg c$).

Definition 3 (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:*

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

$\times$ *represents failure. 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 4 (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* $\forall \vec{x}_1 \in 2^{X_1}$*,* $\pi(\vec{x}_1) = \times$ *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 only valid QBFs have solution policies, it is clear that all QBFs have sound partial policies.

Example 2 $P = \langle 2, \forall, \{a, b\}, \{c\}, (a \vee b) \wedge (a \rightarrow c) \wedge (b \vee c) \rangle$ *has no solution policy. Here is a sound policy for P:*

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

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

Definition 5 (maximal sound policy) *Let π and π' be 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* $\vec{x}_1 \in 2^{X_1}$*, 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* $\vec{x}_k \in 2^{X_k}$*,* $\pi(\vec{x}_k) \sqsupseteq \pi'(\vec{x}_k)$.

[1] λ represents the *empty policy*. The operator ";" represents the sequential composition of policies. $\pi; \lambda$ is typically abbreviated as π.

$\sqsupseteq$ *is a partial preorder (reflexive and transitive relation);* 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 3 $P = \langle 2, \forall, \{a, b\}, \{c\}, (a \vee b) \wedge (a \rightarrow c) \wedge (b \vee c) \rangle$ *has two maximal sound policies:* π *as reported in Example 2, and* π' *identical to* π *except that it maps* $(\neg a, b)$ *to* $\neg c$.

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

Policy Representation

It is essential to distinguish 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). Another representation scheme for total policies consists of circuits (Liberatore 2005): to each existentially quantified variable $x \in X_i$ of a QBF $P = \langle k, q, X_k, ..., X_1, \Phi \rangle$ is associated a circuit C_x whose inputs are all the universally quantified variables Y from $\bigcup_{j=i+1}^{k} X_j$. For each instantiation $\vec{y}$ of those variables, C_x gives the corresponding value of x.

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 $\text{QBF}_{1,\forall}$. The same policy π can be represented in an exponentially more succinct way as the constant function mapping any X_1-instanciation to λ; 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. 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.

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

We now focus on the practical resolution of $\text{SFQBF}_{2,\forall}$, the second function problem for QBFs from $\text{QBF}_{2,\forall}$, which aims at computing a maximal sound policy for a given P. Why the choice of $k = 2$ and $q = \forall$? It is important, before investigating more complex $\text{SFQBF}_{k,q}$ problems, to focus on the problems at 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. 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.

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 6 (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 7 (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* $\pi(\vec{x}) = D_{\mathcal{S}}(\sigma, \vec{x})$ *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 instances of $\text{QBF}_{2,\forall}$ for which any solution policy is injective, as in the following example: $\forall\{x_1, \ldots, x_n\}\exists\{y_1, \ldots, y_n\} \bigwedge_{i=1}^{n}(x_i \leftrightarrow y_i)$.

However, it is possible to encode the solution policies π for the set of QBFs of this example (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[2].

This theorem generalizes Theorem 5 from (Liberatore 2005) in two directions: considering maximally sound policies (instead of the proper subset of it consisting of total policies), and considering any tractable representation scheme (and not only the so-called directional representation scheme as in (Liberatore 2005)).

Given Proposition 3, it seems reasonable 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 8 (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 $D_{\mathcal{S},\sigma}$ such that for any $\vec{x} \in 2^X$, $D_{\mathcal{S},\sigma}$ computes $\pi(\vec{x}) = D_{\mathcal{S},\sigma}(\vec{x})$ in time polynomial in $|\sigma| + |\vec{x}|$.*

The decomposition approach

We present two ways of decomposing the search for a maximally sound policy into easier subproblems. The first one is based on the observation that 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. In order to formalize this, we introduce the notion of *subdecision* and how to merge them.

Definition 9 (subdecision) *An instantiation of* some *(not necessarily all) variables of Y (or equivalently, a satisfiable term γ_Y on Y) is called a* subdecision. *The set of all subdecisions is denoted by 3^Y (as each variable can be mapped to true, false or undefined). A* (partial) subpolicy *for $\forall X \exists Y \Phi$ is a mapping $\pi : 2^X \to 3^Y \cup \{\times\}$ assigning a subdecision (or $\times$) to each X-instantiation. 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*
$$\gamma_Y.\gamma'_Y = \begin{cases} \gamma_Y \wedge \gamma'_Y & \text{if } \gamma_Y \wedge \gamma'_Y \text{ is satisfiable} \\ \times & \text{otherwise} \end{cases}.$$

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

Definition 10 (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;
- *if σ_1 and σ_2 are in* PD, *then $\sigma_1 \odot \sigma_2$ is in* PD.

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

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

We abbreviate (`if` φ `then` σ `else` $\times$) 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 4 *Let $\sigma_1 =$* `if` $x_1 \leftrightarrow x_2$ `then` y_1 `else` $\neg y_1$, *$\sigma_2 =$* `if` x_1 `then` $\neg y_2$, *and $\sigma = \sigma_1 \odot \sigma_2$. The corresponding policies are given by*

	π_{σ_1}	π_{σ_2}	π_σ
(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 5 *A tractable representation in* PD *of the solution policy for*
$\forall x_1 \ldots x_n \exists y_1 \ldots y_n \bigwedge_{i=1}^n (x_i \leftrightarrow y_i)$ *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 represented by the description
$$\sigma = \texttt{Case}\ \varphi_{j_1}^X\texttt{:}\ \vec{y}_{j_1}\texttt{;}\ \ldots\texttt{;}\ \varphi_{j_q}^X\texttt{:}\ \vec{y}_{j_q}\ \texttt{End}$$
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$_{2,\forall}$ 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 (Darwiche 1997; Lang, Liberatore, & Marquis 2002)).

[2] We have also derived the stronger result according to which, under the same assumptions, NP $\subseteq$ P/poly holds; due to space limitations, we refrain from presenting it.

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 a 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).

Example 6 *Consider the instance* $\forall\{x_1, x_2, x_3\} \exists\{y_1, y_2\}\Phi$, *where* $\Phi = (x_1 \vee x_2) \wedge (x_3 \leftrightarrow (y_1 \leftrightarrow y_2))$. *Written as such,* Φ *cannot be decomposed as* $\Phi = \Phi_X \wedge \Phi_Y$, *because* $X = \{x_1, x_2, x_3\}$ *and* $Y = \{y_1, y_2\}$ *are not marginally conditionally independent with respect to* Φ. *However, instanciating* x_3 *breaks the links between* X *and* Y *and gives the following equivalent form for* Φ*:*

$$[(x_3 \wedge (x_1 \vee x_2)) \wedge (y_1 \leftrightarrow y_2)] \vee [(\neg x_3 \wedge (x_1 \vee x_2)) \wedge \neg(y_1 \leftrightarrow y_2)]$$

then Proposition 5 applies and gives, for instance, the following maximal sound policy described compactly by

$$\sigma = \left| \begin{array}{l} \texttt{Case} \\ \quad x_3 \wedge (x_1 \vee x_2) : y_1 \wedge y_2; \\ \quad \neg x_3 \wedge (x_1 \vee x_2) : \neg y_1 \wedge y_2 \\ \texttt{End} \end{array} \right.$$

Note that $\pi_\sigma(\neg x_1, \neg x_2, x_3) = \pi_\sigma(\neg x_1, \neg x_2, \neg x_3) = \times$.

The second way of decomposing a problem into subproblems is based on the observation that 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] The following 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 $\text{SFQBF}_{2,\forall}$ into two (or several, when iterated) instances of $\text{SFQBF}_{2,\forall}$ 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. (Darwiche 2001)).

Example 7 *Consider* $\forall\{x_1, x_2, x_3\}\exists\{y_1, y_2, y_3\}\Psi$, *where* $\Psi = (x_1 \vee x_2) \wedge (x_3 \leftrightarrow (y_1 \leftrightarrow y_2)) \wedge ((x_1 \leftrightarrow x_2) \leftrightarrow y_3)$. $\Psi \equiv \Psi_{X,y_1,y_2} \wedge \Psi_{X,y_3}$, *where* $\Psi_{X,y_1,y_2} = \Phi$ *(from Example 6) and* $\Psi_{X,y_3} = (x_1 \leftrightarrow x_2) \leftrightarrow y_3$. *Then, using the policy description* σ *determined in Example 6 and Proposition 6,* $\sigma' = \sigma \odot (\texttt{if}\ x_1 \leftrightarrow x_2\ \texttt{then}\ y_3\ \texttt{else}\ \neg y_3)$ *is the description of a maximal sound policy for* $\forall\{x_1, x_2, x_3\}\exists\{y_1, y_2, y_3\}\Psi$.

[3] As briefly evoked in (Rintanen 1999a) (Section 6), such independence properties help solving QBF instances.

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 (Darwiche & Marquis 2001)).* σ *is a tractable representation of a maximal sound policy for* P.

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. (Darwiche & Marquis 2001)). 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 e.g., in (Boufkhad *et al.* 1997; Darwiche 2004) 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 $\text{QBF}_{2,\forall}$.

Conclusion and Related Work

The specificities of our work are the following ones: define partial, but maximally sound policies for a QBF Σ, even when it is not valid (other approaches would handle such Σ by concluding that it is impossible to find a solution policy); address the issues of the size of partial policies and their compact representation; focus on the specific problem $\text{QBF}_{2,\forall}$ and show how techniques such as decomposition and compilation can be fruitfully exploited for computing maximally sound policies.

(Kleine-Büning, Subramani, & Zhao 2003) investigate the properties of QBFs having polysize solution policies of a specific kind (e.g., when each $\exists$ variable y is a monotone term – or a boolean constant – built up from $\forall$ variables before y in the prefix). Contrariwise to our work, no restriction is put on the prefix of instances in their study; on the other

hand, it is not the case that every positive QBF (even when from $\mathrm{QBF}_{2,\forall}$) has a polysize solution policy; furthermore, they do not consider partial policies. This shows their approach mainly orthogonal to ours.

(Chen 2004) defines the notion of decomposability of a set of functions using policies in the QCSP framework; the main purpose is to show that if an operation μ is j-collapsible then any constraint language Γ invariant under μ is j-collapsible (Theorem 7), from which tractability results for QCSPs (Γ) are derived. Neither the notion of partial policy nor the problem of their representation are considered in (Chen 2004).

Closer to our work, (Liberatore 2005) considers the representation issue for total policies using a circuit-based representation scheme. The complexity of determining whether a given QBF has a total policy representation, with size bounded by a given integer k is identified and shown hard, even in the case k is in unary notation. In some sense, our work completes (Liberatore 2005) by focusing on partial policies, generalizing some results and focusing on other representation schemes.

References

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

Cadoli, M.; Giovanardi, A.; and Schaerf, M. 1998. An algorithm to evaluate quantified boolean formulae. In *AAAI'98*, 262–267.

Chen, H. 2004. Collapsibility and consistency in quantified constraint satisfaction. In *AAAI-04*, 155–160.

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

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

Darwiche, A. 2001. Decomposable negation normal form. *JACM* 48(4):608–647.

Darwiche, A. 2004. New Advances in Compiling CNF into Decomposable Negation Normal Form. In *ECAI'04*, 328–332.

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

Feldmann, R.; Monien, B.; and Schamberger, S. 2000. A distributed algorithm to evaluate quantified boolean formulas. In *AAAI'00*, 285–290.

Giunchiglia, E.; Narizzano, M.; and Tacchella, A. 2001. Backjumping for quantified boolean logic satisfiability. In *IJCAI'01*, 275–281.

Kleine-Büning, H.; Subramani, K.; and Zhao, X. 2003. On boolean models for quantified boolean Horn formulas. In *SAT'03*, 93–104.

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

Letz, R. 2002. Lemma and Model Caching in Decision Procedures for Quantified Boolean Formulas. In *Tableaux'02*, 160–175.

Liberatore, P. 2005. Complexity issues in finding succinct solutions of pspace-complete problems. Technical report, CS.AI/0503043, Computing Research Repository (CoRR).

Littman, M. 1999. Initial experiments in stochastic satisfiability. In *AAAI'99*, 667–672.

Majercik, S., and Littman, M. 1999. Contingent planning under uncertainty via stochastic satisfiability. In *AAAI'99*, 549–556.

Pan, G., and Vardi, M. 2004. Symbolic Decision Procedures for QBF. In *CP'04*, 453–467.

Rintanen, J. 1999a. Constructing conditional plans by a theorem-prover. *JAIR* 10:323–352.

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

Zhang, L., and Malik, S. 2002. Towards a symmetric treatment of Satisfaction and Conflicts in Quantified Boolean Formula Evaluation. In *CP'02*, 200–215.