Representing Policies for Quantified Boolean Formulae

Sylvie Coste-Marquis, Hélène Fargier, Jérôme Lang, Daniel Le Berre, Pierre Marquis

Representing Policies for Quantified Boolean Formulae

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 Formulae (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, called the matrix of the QBF, together with an ordered partition of its variables, corresponding to quantifier alternations, called the prefix of the QBF. Formally, such a QBF is closed, polite and prenex. Since the validity problem for any QBF can be reduced in polynomial time to the validity problem for a closed, polite and prenex QBF, we focus on such QBFs in the following. For instance, $\exists\{a\}\ \forall\{b,d\}\ \exists\{c\}\ ((a \wedge \neg c) \rightarrow (b \wedge d))$ is a QBF. Any (closed) 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. It is the canonical PSPACE-complete problem.

Solving the decision problem QBF has become for a few years an important research area in AI. Several explanations for this can be provided, including the fact that many AI problems whose complexity is located in PSPACE can be reduced to QBF and then solved by QBF solvers (see e.g., (Egly *et al.* 2000; Fargier, Lang, & Marquis 2000; Rintanen 1999a; Pan, Sattler, & Vardi 2002; Besnard *et al.* 2005)); furthermore, there are some empirical evidences 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 QBF solvers have been developed for the past few years (see among others (Cadoli, Giovanardi, & Schaerf 1998; Rintanen 1999b; Feldmann, Monien, & Schamberger 2000; Rintanen 2001; Giunchiglia, Narizzano, & Tacchella 2001; Letz 2002; Zhang & Malik 2002; Benedetti 2005b; GhasemZadeh, Klotz, & Meinel 2004; Pan & Vardi 2004; Audemard & Saïs 2004) and three QBF evaluations have been organized (Le Berre, Simon, & Tacchella 2003; Le Berre *et al.* 2004; Narizzano, Pulina, & Tacchella 2006).

Obviously, QBFs can be viewed as planning problems under incomplete knowledge and feedback as well as sequential two-player games with complete information. For instance, $\exists\{a\}\forall\{b\}\exists\{c\}\forall\{d\}\Phi$ represents a game with two players $P_\forall$ and $P_\exists$, playing alternatively by assigning a propositional variable: $P_\exists$ starts by assigning a value to a, then $P_\forall$ assigns a value to b, etc. The goal of $P_\exists$ 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 $P_\exists$. When the game is understood as a game against nature (or as a planning problem with nondeterministic actions or exogenous events), instantiations of existentially (respectively universally) quantified variables correspond to plays by the agent (respectively by nature), and winning strategies are policies (or conditional plans). 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 *par-*

*This work is supported by the IUT de Lens, the Région Nord/Pas de Calais through the IRCICA Consortium and the COCOA Project, and by the European Community FEDER Program.

tial policies which 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.

The rest of the paper is organized as follows. First, some formal preliminaries are provided. Then, we define several notions of policies for QBFs: total policies, partial policies, sound and maximal sound policies. Then, we focus on the problem of representing policies, and we present two approaches for representing and exploiting policies for QBFs of the form $\forall X\ \exists Y\ \Phi$. Finally, we discuss some related work before concluding the paper.

Formal Preliminaries

In the following, $PROP_{PS}$ denotes the propositional language built up from a finite set PS of symbols, the usual connectives $\neg$, $\vee$, $\wedge$, $\rightarrow$, $\leftrightarrow$ and the Boolean constants $\top$, $\bot$ in the standard way. $Var(\Sigma)$ is the set of symbols from PS occurring in formula Σ. $\vec{x}$ 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 $\vec{y}$). $\models$ denotes entailment and $\equiv$ denotes equivalence.

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 $\neg x \in \vec{x}$).

Definition 1 (quantified boolean formula) *Let k be a positive integer and $q \in \{\exists, \forall\}$. 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 of propositional variables occurring in $\Phi \in PROP_{PS}$. k is the rank of P and q the first quantifier of its prefix.*

Example 1
$\langle 2, \forall, \{a, b\}, \{c\}, (a \vee b) \wedge (a \rightarrow c) \wedge (b \vee c)\rangle$ *is a QBF.*

Example 2
$\langle 3, \exists, \{a\}, \{b\}, \{c, d\}, (a \rightarrow (c \wedge d)) \wedge (b \leftrightarrow \neg c)\rangle$ *is another QBF.*

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

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

$\text{QBF}_{k,q}$ is the set of all QBFs of rank k and first quantifier q. We also note $last(k, q)$ the innermost quantifier of the considered QBF and $|\exists(k, q)|$ (resp. $|\forall(k, q)|$) the number of groups of existential (resp. universal) quantifiers of the QBF. We now define formally the decision problem QBF through its positive instances:

Definition 2 (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:*

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

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

Example 1 (continued).
$\langle 2, \forall, \{a, b\}, \{c\}, (a \vee b) \wedge (a \rightarrow c) \wedge (b \vee c)\rangle$ *is not a positive instance of* QBF.

Example 2 (continued).
$\langle 3, \exists, \{a\}, \{b\}, \{c, d\}, (a \rightarrow (c \wedge d)) \wedge (b \leftrightarrow \neg c)\rangle$ *is a positive instance of* QBF.

While QBF is the canonical PSPACE-complete problem, $\text{QBF}_{k,\exists}$ (resp. $\text{QBF}_{k,\forall}$) is the canonical Σ_k^p-complete (resp. Π_k^p-complete) problem. In particular, $\text{QBF}_{1,\exists}$ (resp. $\text{QBF}_{1,\forall}$) coincides with the satisfiability problem SAT (resp. the validity problem VAL).

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

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

λ represents the *empty policy*. The operator ";" represents the sequential composition of policies. $\pi; \lambda$ is typically abbreviated as π. $2^{X_k} \rightarrow TP(k-1, \exists, X_{k-1}, ..., X_1)$ denotes the set of all total functions from 2^{X_k} to $TP(k-1, \exists, X_{k-1}, ..., X_1)$.

In particular, 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 $\lambda_{X_1} = \{\vec{x}_1 \mapsto \lambda \mid \vec{x}_1 \in 2^{X_1}\}$ which maps any X_1-instantiation to λ;[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}.

Example 3
A total policy of $TP(3, \exists, \{e, f\}, \{a, b\}, \{c, d\})$ *is:* $\pi = (e, \neg f); \pi'$*, where*

$\pi'((\neg a, \neg b)) = (c, d)$;
$\pi'((\neg a, b)) = (c, d)$;
$\pi'((a, \neg b)) = (\neg c, d)$;
$\pi'((a, b)) = (\neg c, d)$.

Intuitively, performing π *consists in first instantiating* e *to true and* f *to false, then observing the values taken by* a *and* b *and then acting consequently,* i.e.*, instantiating* c *and* d *to true if* a *is false, or* c *to false and* d *to true otherwise.*

Definition 4 (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:*

- $k = 0$, $\pi = \lambda$, *and* $\Phi \equiv \top$ *;*
- $k \geq 1$, $q = \exists$ *and* $\pi = (\vec{x}_k; \pi')$ *with* $\pi' \models \langle k-1, \forall, X_{k-1}, ..., X_1, \Phi_{\vec{x}_k}\rangle$ *;*
- $k \geq 1$, $q = \forall$*, 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 (continued).
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 (continued).
$\langle 3, \exists, \{a\}, \{b\}, \{c, d\}, (a \rightarrow (c \wedge d)) \wedge (b \leftrightarrow \neg c)\rangle$ *is satisfied by*

$$\pi = \neg a; \left[\begin{array}{ll} (b) & \mapsto (\neg c, d) \\ (\neg b) & \mapsto (c, d) \end{array}\right].$$

The following result shows how positive instances of QBF and total policies for QBFs are related:

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*.*

This result is nothing really new and is proven easily; we mention it because it enables us to define formally the function problem FQBF 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* $\text{FQBF}_{k,q}$ *for* P *consists in finding a total policy* π *such that* $\pi \models P$*, if there exists any, and in stating that no solution policy exists for* P *otherwise. We note* FQBF *the function problem associated to any QBF.*

[1] $\vec{x}_1 \mapsto \lambda$ is another notation for $(\vec{x}_1, \lambda)$ which reflects in a more salient way that λ_{X_1} is a function (anyway, a function is a set of pairs).

Now, asking for a solution policy is often too much demanding. For instance, let us consider the following example.

Example 4
$P = \langle 2, \forall, \{a, b\}, \{c\}, (a \rightarrow c) \wedge (b \rightarrow \neg c)\rangle$ *is not valid because the instantiation* (a, b) *makes* Φ *unsatisfiable. If* P *is understood as a game against nature, instantiations of existentially (respectively universally) quantified variables correspond to plays by the agent (respectively by nature), and winning strategies are policies (or conditional plans). 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 undefined for some possible instantiations of groups of universally quantified variables, hence we call them *partial policies*:

Definition 6 (partial policy) *The set* $PP(k, q, X_k, ..., X_1)$ *of* partial policies *for QBFs from* $\text{QBF}_{k,q}$ *is defined inductively as follows:*

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

$\times$ 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.

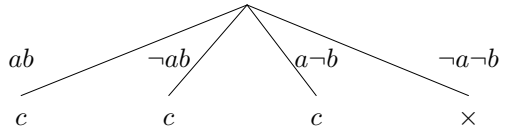
Observe that the set of partial policies for a given $\text{QBF}_{k,q}$ includes the set of total policies for that class of QBFs and is finite.

With each partial policy π we can associate a tree T_π: if π is a policy of $PP(k, q, X_k, ..., X_1)$ then T_π is the tree whose depth is equal to $|\forall(k, q)|$, whose leaves are labelled by $\times$, λ or by instantiations of X_1 if $last(k, q) = \exists$ whose internal nodes (except the root) are labelled by instantiations of groups of existentially quantified variables, and whose edges are labelled by instantiations of groups of universally quantified variables. Such a tree representation for a partial policy of example 1 is:

Example 1 (continued).
A partial policy π_1 *for* $P = \langle 2, \forall, \{a, b\}, \{c\}, (a \vee b) \wedge (a \rightarrow c) \wedge (b \vee c)\rangle$ *is:*

$$\pi_1 = \left[\begin{array}{ll} (a, b) & \mapsto c \\ (\neg a, b) & \mapsto c \\ (a, \neg b) & \mapsto c \\ (\neg a, \neg b) & \mapsto \times \end{array}\right]$$

and can be represented by the tree:



Clearly enough, one is not interested in any partial policy for a given QBF, but only in sound ones:

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. $k = 0$ *and* ($\pi = \times$ *or* ($\pi = \lambda$ *and* $\varphi \equiv \top$));
2. $q = \exists$ *and* $\pi = \times$;
3. $(k, q) = (1, \exists)$, $\pi = \vec{x}_1$ *and* $\vec{x}_1 \models \Phi$;
4. $(k, q) = (1, \forall)$ *and* $\forall \vec{x}_1 \in 2^{X_1}$, $\pi(\vec{x}_1) = \times$ *or* ($\pi(\vec{x}_1) = \lambda$ *and* $\vec{x}_1 \models \Phi$);
5. $k > 1$, $q = \exists$, $\pi = \vec{x}_k; \pi_{k-1}$ *and* π_{k-1} *is sound for* $\langle k-1, \forall, X_{k-1}, ..., X_1, \Phi_{\vec{x}_k}\rangle$;
6. $k > 1$, $q = \forall$, *and for any* $\vec{x}_k \in 2^{X_k}$, $\pi(\vec{x}_k)$ *is sound for* $\langle k-1, \exists, X_{k-1}, ..., X_1, \Phi_{\vec{x}_k}\rangle$.

Equivalently, a policy is sound if and only if on every path of its associated tree where $\times$ does not appear, the variable assignments along this path (on its nodes and branches) make Φ true.

Example 1 (continued).
The policy π_1 given above is sound. Another sound policy for P is:

$$\pi_2 = \begin{bmatrix} (a,b) & \mapsto \times \\ (\neg a, b) & \mapsto \times \\ (a, \neg b) & \mapsto \times \\ (\neg a, \neg b) & \mapsto \times \end{bmatrix}$$

Contrastingly, the following partial policy for P is not sound:

$$\pi_3 = \begin{bmatrix} (a,b) & \mapsto c \\ (\neg a, b) & \mapsto c \\ (a, \neg b) & \mapsto c \\ (\neg a, \neg b) & \mapsto c \end{bmatrix}$$

While only valid QBFs have solution policies, it is clear that every QBF $P = \langle k, q, X_k, ..., X_1, \Phi\rangle$ has a sound partial policy. Indeed, if $q = \exists$ then $\pi = \times$ is a sound policy for P, while if $q = \forall$, then the policy π given by $\pi(\vec{x}_k) = \times$ for every $\vec{x}_k \in 2^{X_k}$ is a sound policy for P.

Intuitively, the best policies among the sound ones are those built up from internal policies where $\times$ is used when needed, only:

Definition 8 (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:*

- $k = 0$ *and* $\pi = \lambda$;
- $q = \exists$ *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* $\vec{x}_k \in 2^{X_k}$, $\pi(\vec{x}_k) \sqsupseteq \pi'(\vec{x}_k)$.

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 $\pi \not\sqsupseteq \pi'$.*

It is easy to show that the covering relation $\sqsupseteq$ is a partial order (i.e., a reflexive and transitive relation over the set of partial policies).

Maximal sound policies minimize failure: a policy is maximal sound if and only if each time $\times$ appears on a path of its associated tree, the variable assignments along this path (on its nodes and branches) falsify Φ (whatever the values of the unassigned variables); in other words, there is no assignment of the existentially quantified variables at this node for which there would still be a hope of seeing Φ eventually satisfied.

Example 1 (continued).
$\langle 2, \forall, \{a, b\}, \{c\}, (a \vee b) \wedge (a \rightarrow c) \wedge (b \vee c)\rangle$ has two maximal sound policies: π_1 as reported before, and π'_1 identical to π except that it maps $(\neg a, b)$ to $\neg c$.

Clearly, every QBF P has at least one maximal sound policy (just because it has at least one sound policy – and the fact that the set of all partial policies for P is finite); furthermore, if a solution policy for P exists, then solution policies and maximal sound policies coincide.

Especially, every positive instance of $P = \langle \exists, X_1, \Phi\rangle$ of $\text{QBF}_{1,\exists}$ has as many maximal sound policies as Φ has models, while every positive instance from $\text{QBF}_{1,\forall}$ has a unique maximal sound policy, namely λ_{X_1}. Every negative instance of $\text{QBF}_{1,\exists}$ has a unique maximal sound policy, namely $\times$. Every negative instance of $\text{QBF}_{1,\forall}$ has a unique maximal sound policy π such that: for every $\vec{x_1} \in 2^{X_1}$, if $\vec{x_1} \models \Phi$, then $\pi(\vec{x_1}) = \lambda$, else $\pi(\vec{x_1}) = \times$.

In the following, we are mainly interested in the representation issue for the solutions of the function problems FQBF and SFQBF:

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. We note* SFQBF *the second function problem associated to any QBF.*

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 $\pi(\vec{x})$ of an instantiation $\vec{x}$ by a given policy for a QBF with first quantifier $\forall$ 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 of a policy π is its tree representation T_π. Observe that the tree representation of a policy is equivalent to the explicit one in the sense that there exists a polytime algorithm which turns the explicit representation into the tree one, and the converse also holds.

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 consists of (a) the explicit representation of a policy $\pi \in TP(k,\ q,\ X_k,\ ...,\ X_1)$ and (b) 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.*

Proof: The proof is by induction on k. The base case is for $k = 0$. π is a solution policy for $P = \langle 0, q, \Phi\rangle$ if and only if Φ is valid. Since $Var(\Phi) = \emptyset$, checking whether Φ is valid can be done in polynomial time in the size of Φ. Now assume that the property holds for every $k = i \geq 0$. Let us show that it holds for $k = i + 1$. Let $P = \langle i + 1, q, X_{i+1}, ..., X_1, \Phi\rangle$ be a QBF. If $q = \exists$, then $\pi = \vec{x}_{i+1}; \pi_i$ with $\pi_i \in TP(i, \forall, X_i, ..., X_1)$ satisfies P if and only if π_i satisfies $\langle i, \forall, X_i, ..., X_1, \Phi_{\vec{x}_{i+1}}\rangle$. Since $\Phi_{\vec{x}_{i+1}}$ can be computed in time polynomial in the size of Φ and X_{i+1}, the induction hypothesis leads directly to the expected conclusion. Finally, if $q = \forall$, then π satisfies P if and only if for every $\vec{x}_{i+1} \in 2^{X_{i+1}}$, the policy $\pi(\vec{x}_{i+1})$ from $TP(i, \exists, X_i, ..., X_1)$ satisfies $\langle i, \exists, X_i, ..., X_1, \Phi_{\vec{x}_{i+1}}\rangle$. Since $\pi = \{(\vec{x}_{i+1}, \pi(\vec{x}_{i+1})) \mid \vec{x}_{i+1} \in 2^{X_{i+1}}\}$, the induction hypothesis completes the proof. ■

For every positive 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 can be 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 positive 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 $\{(\vec{x}_1 \mapsto \lambda) \mid \vec{x}_1 \in 2^{X_1}\}$; again, this representation is a certificate for $\text{QBF}_{1,\forall}$. The same policy π can be represented in an exponentially more succinct way by the name λ_{X_1} of the constant function mapping any X_1-instanciation to λ; obviously, such a (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.

Now, a policy is a function. Instead of representing a policy using a data structure, we can also represent it using *an algorithm which computes the function*. Accordingly, an algorithm is said to represent a solution policy for a QBF $P = \langle k,\ q,\ X_k,\ ...,\ X_1, \Phi\rangle$ if and only if given an instantiation of all universally quantified variables of P it enables to compute an instantiation of all existentially quantified variables of P such that the concatenation of the two instantiations is a model of Φ.

Interestingly, there exist polytime (algorithmic) representations of solution policies for some QBFs. In order to present one of them, we first need the following notions. A propositional fragment $\mathcal{F} \subseteq PROP_{PS}$ is said to enable polytime conditioning (Darwiche & Marquis 2001) (resp. polytime quantification elimination) if and only if there exists a polytime algorithm which, for any $\vec{x} \in 2^X$ with $X \subseteq PS$ and any $\Phi \in \mathcal{F}$ computes a formula from $\mathcal{F}$ equivalent to $\Phi_{\vec{x}}$ (resp. there exists a polytime algorithm which, for any $X \subseteq PS$, any quantifier q and any $\Phi \in \mathcal{F}$ computes a formula from $\mathcal{F}$ equivalent to $qX\Phi$). A propositional fragment $\mathcal{F} \subseteq PROP_{PS}$ is said to enable polytime model finding (i.e. the function problem for SAT) if and only if there exists a polytime algorithm which, for any $\Phi \in \mathcal{F}$ computes a model of Φ if Φ is satisfiable, and returns that no model exists otherwise.

Such fragments are valuable when the purpose is to decide QBF and, more generally, to represent solution policies for QBFs:

Proposition 3 *Let $P = \langle k,\ q,\ X_k,\ ...,\ X_1, \Phi\rangle$ be a QBF where Φ belongs to a propositional fragment $\mathcal{F}$ enabling polytime conditioning, polytime model finding and polytime quantification elimination. Then deciding whether P is a positive instance of* QBF *can be achieved in polynomial time. Furthermore, there exist a polytime (algorithmic) representation of a solution policy for each valid P.*

Proof: In order to decide whether P is valid, it is sufficient to eliminate in polynomial time each quantification of the prefix of P (from the innermost to the outermost qX_k), in Φ within the fragment and to check in polynomial time whether or not the resulting formula has a model. If P is valid and $k = 0$, then we just return the policy $\pi = \lambda$. If P is valid and its first quantifier q is existential, then it is sufficient to remove in polynomial time all the quantifications of the prefix of P in Φ from the innermost to the outermost, but the first one qX_k, then to search in polynomial time for a model of the resulting formula: each such model $\vec{x}_k$ is the first element of a solution policy for P and the converse also holds. If P is valid and its first quantifier q is universal, then given an $X_b k$-instantiation $\vec{x}_k$, it is sufficient to compute in polynomial time the first element of $\pi(\vec{x}_k)$ where π is a solution policy for P. This can be done by considering the QBF $\langle k-1, \exists, X_{k-1} k, ..., X_1, \Phi_{\vec{x}_k}\rangle$. A straightforward induction completes the proof. ■

In (Coste-Marquis *et al.* 2005), some of us have shown that the fragment $\text{OBDD}_<$ of ordered binary decision diagrams enable polytime quantification elimination for the quantification bearing on the last variable w.r.t. $<$. Since $\text{OBDD}_<$ was already known to satisfy polytime conditioning and model finding, this is sufficient to take advantage of the approach described above to represent solution policies for valid QBFs whose matrix is from $\text{OBDD}_<$ and whose prefix is compatible w.r.t.$<$.

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 from $\text{QBF}_{2,\forall}$. 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 $k = 1$ for $\text{QBF}_{k,q}$ has received an enormous attention since it corresponds to the satisfiability problem and the validity problem (depending on q) for propositional formulae. However, it is not very interesting from the point of view of policy representation, as shown before. $\text{SFQBF}_{2,\exists}$ 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.

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 $\pi(\vec{x}) = D_{\mathcal{S}}(\sigma, \vec{x})$ 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 (from (Fargier, Lang, & Marquis 2000)):

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 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 4 *If a polynomially compact and tractable representation scheme $\mathcal{S}$ for maximal sound policies for $QBF_{2,\forall}$ exists, then* NP $\subseteq$ P/poly *holds.*

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 $\pi = I_{\mathcal{S}}(\sigma)$ for P and such that $D_{\mathcal{S}}$ computes $\pi(\vec{x}) = D_{\mathcal{S}}(\sigma, \vec{x})$ for any $\vec{x} \in 2^X$.

Let us associate to any CNF formula Σ of $PROP_{PS}$ such that $Var(\Sigma) = \{p_1, \ldots, p_n\}$ the following instance P_Σ of $\text{QBF}_{2,\forall}$:

$$\forall L \exists V (\Sigma \wedge \bigwedge_{i=1}^{n} (\neg l_i^+ \vee p_i) \wedge (\neg l_i^- \vee \neg p_i))$$

with $V = Var(\Sigma)$, $L = \bigcup_{i=1}^{n}\{l_i^+, l_i^-\} \subseteq PS \setminus V$. Observe that the size of P_Σ is linear in the size of Σ.

Now, if $\mathcal{S}$ is tractable, then there exists a polytime algorithm for deciding clausal entailment from Σ. Indeed, for any clause γ built upon the variables of V, we have $\Sigma \models \gamma$ if and only if $\Sigma \wedge \neg\gamma$ is unsatisfiable. Each possible satisfiable term $\neg\gamma$ corresponds to a vector $\vec{l}$ of 2^L such that $\forall p_i \in V$, (1) if $p_i \in \neg\gamma$ then $l_i^+ \in \vec{l}$ and $\neg l_i^- \in \vec{l}$, (2) if $\neg p_i \in \neg\gamma$ then $\neg l_i^+ \in \vec{l}$ and $l_i^- \in \vec{l}$, and (3) if $p_i \notin \neg\gamma$ and $\neg p_i \notin \neg\gamma$ then $\neg l_i^+ \in \vec{l}$ and $\neg l_i^- \in \vec{l}$. Therefore, for any non-valid clause γ on V, we have $\Sigma \models \gamma$ if and only if $(\Sigma \wedge \bigwedge_{i=1}^{n}(\neg l_i^+ \vee p_i) \wedge (\neg l_i^- \vee \neg p_i))_{\vec{l}}$ is unsatisfiable. Accordingly, given any maximal sound policy π for P_Σ and any non-valid clause γ on V, we have $\Sigma \models \gamma$ if and only if $\pi(\vec{l}) = \times$.

Let $\sigma = R_{\mathcal{S}}(P_\Sigma)$. Since $R_{\mathcal{S}}$ is polysize, the mapping $\Sigma \mapsto R_{\mathcal{S}}(P_\Sigma)$ also is polysize. If the representation scheme $\mathcal{S}$ for π were tractable then checking whether $\pi(\vec{l}) = \times$ could be done in polynomial time (just check whether $D_{\mathcal{S}}(\sigma, \vec{l}) = \times$), therefore deciding whether $\Sigma \models \gamma$ could also be achieved in polynomial time. As a consequence, we would get NP $\subseteq$ P/poly (Selman & Kautz 1996). ■

The inclusion NP $\subseteq$ P/poly is considered very unlikely, because it implies that the polynomial hierarchy collapses at the second level (Karp & Lipton 1980). Actually, the fact that the existence of a polynomially compact and tractable representation scheme $\mathcal{S}$ for maximal sound policies for $\text{QBF}_{2,\forall}$ exists entails that the polynomial hierarchy collapses at the second level can also be proved in a more direct way. Indeed, 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 $\pi(\vec{x}) = D_{\mathcal{S}}(\sigma, \vec{x})$ for any $\vec{x} \in 2^X$. Let us now consider the following nondeterministic algorithm for solving $\text{QBF}_{2,\forall}$:

```
Input:  a QBF P = ⟨2, ∀, X, Y, Φ⟩.
1. guess σ = R_S(P);
2. check that I_S(σ) 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 $\text{QBF}_{2,\forall}$ 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 $\text{QBF}_{2,\forall}$ is in Σ_2^p, hence $\Sigma_2^p = \Pi_2^p$ and the polynomial hierarchy collapses at the second level.

Proposition 4 generalizes Theorem 5 from (Liberatore 2005) in two directions: considering maximal sound policies (instead of the proper subset of it consisting of solution policies), and considering any tractable representation scheme (and not only the so-called directional representation scheme as in (Liberatore 2005)).

Given Proposition 4, 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 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 $D_{\mathcal{S},\sigma}$ 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

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.[2]

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 $\pi : 2^X \to 3^Y$ 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*
$$\gamma_Y.\gamma'_Y = \begin{cases} \gamma_Y \wedge \gamma'_Y & \text{if } \gamma_Y \wedge \gamma'_Y \text{ is satisfiable} \\ \times & \text{otherwise} \end{cases}.$$

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*
$$\texttt{if } \varphi_X \texttt{ then } \sigma_1 \texttt{ else } \sigma_2$$
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 $\pi = I_{PD}(\sigma)$ 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* $I_{PD}(\times)(\vec{x}) = \times$;
- $I_{PD}(\gamma_Y)(\vec{x}) = \gamma_Y$;
- $I_{PD}(\texttt{if } \varphi_X \texttt{ then } \sigma_1 \texttt{ else } \sigma_2)(\vec{x})$
$$= \begin{cases} I_{PD}(\sigma_1)(\vec{x}) & \text{if } \vec{x} \models \varphi_X \\ I_{PD}(\sigma_2)(\vec{x}) & \text{if } \vec{x} \models \neg\varphi_X \end{cases}$$
- $I_{PD}(\sigma_1 \odot \sigma_2) = I_{PD}(\sigma_1) \odot I_{PD}(\sigma_2)$.[3]

To simplify notations,
$$\texttt{if } \varphi \texttt{ then } \sigma \texttt{ else } \times$$
is abbreviated into
$$\texttt{if } \varphi \texttt{ then } \sigma$$
and
```
if φ1 then σ1
     else if φ2 then σ2
                else ...
                     else if φn then σn
```
is abbreviated into Case φ_1: σ_1; ... φ_n: σ_n End

Example 6
Let $P = \forall\{x_1, x_2\}\exists\{y_1, y_2\}\ \Phi$, where
$$\Phi = x_1 \wedge \neg y_2 \wedge (y_1 \leftrightarrow (x_1 \leftrightarrow x_2))$$
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 $\sigma = \sigma_1 \odot \sigma_2$.

[2] As briefly evoked in (Rintanen 1999a) (Section 6), such independence properties can also prove helpful in the practical solving of instances of QBF.

[3] We slightly abuse notations here, using $\odot$ both for merging policies and for merging policy descriptions.

The corresponding policies $\pi_1 = I_{PD}(\sigma_1)$, $\pi_2 = I_{PD}(\sigma_2)$ and $\pi = \pi_1 \odot \pi_2 = I_{PD}(\sigma_1 \odot \sigma_2)$ are given by

	π_1	π_2	π
(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	$\times$	$\times$

π_1 is a subpolicy for P; π_2 is a partial subpolicy for P. We can check that π is a maximal sound policy for P.

Proposition 5 PD *is a tractable representation scheme for maximal sound policies for QBF$_{2,\forall}$.*

Proof: It is clear from its definition that for all $\vec{x} \in 2^X$, $I_{PD}(\vec{x})$ is computable in polynomial time; let D_{PD} be the algorithm that computes $I_{PD}(\sigma)(\vec{x})$ from σ and $\vec{x}$. D_{PD} computes $\pi(\vec{x}) = D_{PD}(\sigma, \vec{x})$ for any $\vec{x} \in 2^X$, where $\pi = I_{PD}(\sigma)$. Therefore, *PD* is a tractable representation scheme for maximal sound policies for QBF$_{2,\forall}$. ■

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

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

Note, however, that *PD* is not necessarily polynomially compact.

We first establish the following, which tells precisely when a partial policy for $\forall X \exists Y \Phi$ is sound.

Proposition 6 *Let $P = \forall X \exists Y \Phi$.*

1. *A partial policy π for P is sound for P if and only if for every $\vec{x} \in 2^X$, $\pi(\vec{x}) = \vec{y} \neq \times$ implies $(\vec{x}, \vec{y}) \models \Phi$.*
2. *A sound policy π for P is maximal sound for P if and only if for every $\vec{x} \in 2^X$, $\pi(\vec{x}) = \times$ implies $\vec{x} \models \neg\Phi$.*

Proof: Remark first that $\pi \in PP(2, \forall, X, Y)$ if and only if, by definition, π is a mapping from 2^X to $2^Y \cup \{\times\}$.

1. Applying point 5 of Definition 7 leads to the equivalent formulation: π is sound for P if and only if for all $\vec{x} \in 2^X$, $\pi(\vec{x})$ is sound for $\exists Y.\Phi_{\vec{x}}$. Now, again after Definition 7, $\pi(\vec{x})$ is sound for $\exists Y.\Phi_{\vec{x}}$ if and only if either $\pi(\vec{x}) = \times$ or $\pi(\vec{x}) \models \Phi_{\vec{x}}$. Putting things together we have that π is sound for P if and only if for all $\vec{x} \in 2^X$, either $\pi(\vec{x}) = \times$ or $(\vec{x}, \pi(\vec{x})) \models \Phi$.
2. Let π be a sound policy for P.

   $\Rightarrow$ Assume that there exists an $\vec{x}^* \in 2^X$ such that $\pi(\vec{x}^*) = \times$ and $\vec{x}^* \not\models \neg\Phi$, which means that there exists a $\vec{y}^* \in 2^Y$ such that $(\vec{x}^*, \vec{y}^*) \models \Phi$. We now build the following partial policy π' for P:

   $$\text{for every } \vec{x} \in 2^X, \pi'(\vec{x}) = \begin{cases} \pi(\vec{x}) & \text{if } \vec{x} \neq \vec{x}^* \\ \vec{y}^* \text{if } \vec{x} = \vec{x}^* \end{cases}$$

   We have immediately that $\pi' \sqsupseteq \pi$ holds, but not $\pi \sqsupseteq \pi'$, that is, $\pi' \sqsupset \pi$. Moreover, π' is sound for P. Therefore, π is not maximal sound for P.

   $\Leftarrow$ Assume that π is not maximal sound for P, which means that there exists a sound partial policy π' such that π' strictly covers π, i.e., $\pi' \sqsupset \pi$. The fact that $\pi' \sqsupset \pi$ means that
   (a) for every $\vec{x} \in 2^X$, $\pi'(\vec{x}) = \times$ implies $\pi(\vec{x}) = \times$, and
   (b) there exists an $\vec{x}^* \in 2^X$ such that $\pi'(\vec{x}^*) \neq \times$ and $\pi(\vec{x}^*) = \times$. Consider this $\vec{x}^*$. Since $\pi'(\vec{x}^*) \neq \times$, there is a $\vec{y}^*$ such that $\pi'(\vec{x}^*) = \vec{y}^*$, and since π' is sound for P, it must be the case that
   $$(1)\ (\vec{x}^*, \vec{y}^*) \models \Phi$$
   Therefore, $\vec{x}^* \not\models \neg\Phi$, that is, there is a $\vec{x} \in 2^X$ such that $\pi(\vec{x}) = \times$ and $\vec{x} \not\models \neg\Phi$.

■

Putting points (1) and (2) together, and after some rewriting, we have:

Corollary 1 *π is maximal sound for P if and only if for all $\vec{x} \in 2^X$, either*
(a) $\pi(\vec{x}) = \gamma_Y$ for some $\gamma_Y \in 3^Y$, and $(\vec{x}, \gamma_Y) \models \Phi$, or
(b) $\pi(\vec{x}) = \times$ and there is no $\gamma_Y \in 3^Y$ s. t. $(\vec{x}, \gamma_Y) \models \Phi$.

Proposition 7 *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 $2p$ formulae such that*

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

Let $J = \{j \mid \varphi_j^Y \text{ is satisfiable}\} = \{j_1, \ldots, j_q\}$ 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.

Proof: First, note that $\bigvee_{i=1}^p (\varphi_i^X \wedge \varphi_i^Y)$ is logically equivalent to $\bigvee_{i \in J} (\varphi_i^X \wedge \varphi_i^Y)$, because $i \notin J$ implies that $\varphi_i^X \wedge \varphi_i^Y$ is unsatisfiable. Therefore, without loss of generality, we can assume that $J = \{1, \ldots, q\}$, that is, for all $i \in \{1, \ldots, q\}$, φ_j^Y is satisfiable. For every $j \in \{1, \ldots, p\}$, let $\vec{y}_i$ such that $\vec{y}_i \models \varphi_j^Y$. Now, let

$$\sigma = \texttt{Case } \varphi_1^X\texttt{: } \vec{y}_1\texttt{; } \ldots\texttt{; } \varphi_q^X\texttt{: } \vec{y}_q \texttt{ End}$$

For all $\vec{x}$, let $f(\vec{x}) = \min\{i \mid \vec{x} \models \varphi_i^X\}$, with the convention $f(\vec{x}) = \infty$ if $\vec{x} \models \neg\varphi_i^X$ for all $i = 1, \ldots, p$. Then we have $I_{PD}(\sigma)(\vec{x}) = \vec{y}_{f(\vec{x})}$ if $f(\vec{x}) \neq \infty$, and $I_{PD}(\sigma)(\vec{x}) = \times$ otherwise. Using Proposition 6, this immediately shows that $I(\sigma)$ is sound for P.

Now, suppose that $I_{PD}(\sigma)$ is not a maximal sound policy for P. Then there exists a sound policy π for P strictly covering $I_{PD}(\sigma)$, i.e., such that $\pi \sqsupset I_{PD}(\sigma)$. The fact that $\pi \sqsupset I_{PD}(\sigma)$ means that
(a) for every $\vec{x} \in 2^X$, $\pi(\vec{x}) = \times$ implies $I_{PD}(\sigma)(\vec{x}) = \times$, and
(b) there exists an $\vec{x}^* \in 2^X$ such that $\pi(\vec{x}^*) \neq \times$ and $I_{PD}(\sigma)(\vec{x}^*) = \times$. Consider this $\vec{x}^*$. Since $\pi(\vec{x}^*) \neq \times$, there is a $\vec{y}^*$ such that $\pi(\vec{x}^*) = \vec{y}^*$, and since π is sound for P, it must be the case that

$$(1)\ (\vec{x}^*, \vec{y}^*) \models \Phi$$

Now, $I(\sigma)(\vec{x}^*) = \times$ implies that $f(\vec{x}^*) = \infty$, that is,

$$(2)\ \vec{x}^* \models \neg\varphi_1^X \wedge \ldots \wedge \neg\varphi_p^X$$

Since Φ is logically equivalent to $\bigvee_{i=1}^p (\varphi_i^X \wedge \varphi_i^Y)$, (2) implies $\vec{x}^* \models \neg\Phi$, which contradicts (1). Henceforth, $I(\sigma)$ is a maximal sound policy for P. ■

The interest of Proposition 7 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 (Darwiche 1997; Lang, Liberatore, & Marquis 2002)).

Example 7
Let $P = \forall\{a,b\}\exists\{c,d\}\Phi$, where

$$\Phi = ((a \leftrightarrow b) \wedge c) \vee (a \wedge \neg c \wedge d))$$

The description of a maximal sound policy for P is

$$\sigma = \texttt{Case}\ (a \leftrightarrow b) : (c,d);\ a : (\neg c, d)\ \texttt{End}$$

The associated policy $I_{PD}(\sigma)$ is

$$I_{PD}(\sigma) = \begin{bmatrix} (a,b) & \mapsto (c,d) \\ (\neg a, b) & \mapsto \times \\ (a, \neg b) & \mapsto (\neg c, d) \\ (\neg a, \neg b) & \mapsto (c,d) \end{bmatrix}$$

Furthermore, Proposition 7 immediately tells how to compute a maximal sound policy in polynomial time for $\forall X \exists Y \Phi$ when Φ is in DNF, which implies the following:

Corollary 2 *When Φ is a DNF formula, a maximal sound policy for $\forall X \exists Y \Phi$ is computable in polynomial time.*

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 a positive instance (the latter is 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 those subpolicies.

Proposition 8 *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 formulae Φ_1 and Φ_2 of respectively $PROP_{X \cup Y_1}$ and $PROP_{X \cup Y_2}$ such that $\Phi \equiv \Phi_1 \wedge \Phi_2$. 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, maximal sound for $\forall X \exists Y \Phi_1$ and $\forall X \exists Y \Phi_2$ respectively, such that $\pi = \pi_1 \odot \pi_2$.*

Proof: Assume $\Phi \equiv \Phi_1 \wedge \Phi_2$, where Φ_1 and Φ_2 are respectively in $PROP_{X \cup Y_1}$ and $PROP_{X \cup Y_2}$.

$\Rightarrow$ Let π be a maximal sound policy for $\exists X \forall Y \Phi_1$. Define now π_1 as follows:

1. if $\pi(\vec{x}) \neq \times$ then $\pi_1(\vec{x})$ is the restriction of π on Y_1;
2. if $\pi(\vec{x}) = \times$ and $\vec{x} \not\models \neg\Phi_1$ then $\pi_1(\vec{x}) = \vec{y}_1$ for some $\vec{y}_1$ such that $(\vec{x}, \vec{y}_1) \models \Phi_1$;
3. if $\pi(\vec{x}) = \times$ and $\vec{x} \models \neg\Phi_1$ then $\pi_1(\vec{x}) = \times$;

and similarly for π_2, replacing Y_1 and Φ_1 by, respectively, Y_2 and Φ_2. π_1 (resp. π_2) is a policy for $\forall X \exists Y \Phi_1$ (resp. $\forall X \exists Y \Phi_2$). We first check that $\pi = \pi_1 \odot \pi_2$. Indeed:

– if $\pi(\vec{x}) = \times$ then, since π is maximal sound, we have $\vec{x} \models \neg\Phi$. Now, assume that $\pi_1 \odot \pi_2(\vec{x}) \neq \times$. Then $\pi_1(\vec{x}) \neq \times$ and $\pi_2(\vec{x}) \neq \times$, which implies that there exists $\vec{y}_1 \in 2^{Y_1}$ and $\vec{y}_2 \in 2^{Y_2}$ such that $(\vec{x}, \vec{y}_1) \models \Phi_1$ and $(\vec{x}, \vec{y}_2) \models \Phi_2$, because π_1 and π_2 are sound. The latter implies that there exists $\vec{y}_1 \in 2^{Y_1}$ and $\vec{y}_2 \in 2^{Y_2}$ such that $(\vec{x}, \vec{y}_1, \vec{y}_2) \models \Phi_1 \wedge \Phi_2$, i.e., $(\vec{x}, \vec{y}_1, \vec{y}_2) \models \Phi$, which contradicts $\vec{x} \models \neg\Phi$. Hence, $\pi_1 \odot \pi_2(\vec{x}) \neq \times$.

Then, we check that π_1 and π_2 are maximal sound for, respectively, $\forall X \exists Y \Phi_1$ and $\forall X \exists Y \Phi_2$. The construction of π_1 ensures that $\vec{x} \models \neg\Phi_1$ holds whenever $\pi_1(\vec{x}) = \times$, which, by Proposition 6, implies that π_1 is maximal and sound. The proof for π_2 is similar.

$\Leftarrow$ Assume there exist two subpolicies π_1, π_2, maximal sound for $\forall X \exists Y \Phi_1$ and for $\forall X \exists Y \Phi_2$ respectively, such that $\pi = \pi_1 \odot \pi_2$.
(a) Let us first show that π is sound. Let $\vec{x} \in 2^X$ such that $\pi(\vec{x}) \neq \times$. Since $\pi(\vec{x}) \neq \times$, we have $\pi(\vec{x}) = \vec{y}$ for some $\vec{y} \in 2^Y$. Let $\vec{y}_1$ and $\vec{y}_2$ be the projections of $\vec{y}$ on Y_1 and Y_2, respectively. Since $\pi(\vec{x}) = \pi_1(\vec{x}).\pi_2(\vec{x})$, we necessarily have $\pi_1(\vec{x}) = \vec{y}_1$ and $\pi_2(\vec{x}) = \vec{y}_2$. Now, π_1 is sound for $\forall X \exists Y \Phi_1$, therefore, by Proposition 6, (1) $(\vec{x}, \vec{y}_1) \models \Phi_1$. Similarly, (2) $(\vec{x}, \vec{y}_2) \models \Phi_2$. (1) and (2) imply that $(\vec{x}, \vec{y}) \models \Phi_1 \wedge \Phi_2$, that is, $(\vec{x}, \vec{y}) \models \Phi$. This being true for all $\vec{x} \in 2^X$ such that $\pi(\vec{x}) \neq \times$, we conclude that π is sound.
(b) Let us show now that π is maximal sound. Suppose it is not. Then, by Proposition 6, there exists $\vec{x} \in 2^X$ such that $\pi(\vec{x}) = \times$ and $\vec{x} \not\models \neg\Phi$. $\vec{x} \not\models \neg\Phi$ is equivalent to the existence of $\vec{y} \in 2^Y$ such that $(\vec{x}, \vec{y}) \models \Phi$. Let $\vec{y}_1$ and $\vec{y}_2$ be the projections of $\vec{y}$ on Y_1 and Y_2, respectively. Then $(\vec{x}, \vec{y}) = (\vec{x}, \vec{y}_1, \vec{y}_2) \models \Phi$, therefore $(\vec{x}, \vec{y}_1) \models \Phi_1$ and $(\vec{x}, \vec{y}_2) \models \Phi_2$, i.e.,

(1) $\vec{x} \not\models \neg\Phi_1$ and $\vec{x} \not\models \neg\Phi_2$.

Now, since $\pi = \pi_1 \odot \pi_2$, $\pi(\vec{x}) = \times$ implies that either $\pi_1(\vec{x}) = \times$ or $\pi_2(\vec{x}) = \times$. Without loss of generality, assume $\pi_1(\vec{x}) = \times$. This, together with (1) and Proposition 6, implies that π_1 is not maximal sound for $\forall X \exists Y_1 \Phi_1$, a contradiction.

■

Proposition 8 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 8
Let $P = \langle \forall\{a,b\}\exists\{c,d,e\}\Phi$, *where*

$$\Phi = (((a \leftrightarrow b) \wedge c) \vee (a \wedge \neg c \wedge d)) \wedge ((a \vee b) \leftrightarrow \neg e)$$

$\{c,d\}$ *and* $\{e\}$ *are independent given* $\{a,b\}$ *given* Φ*: indeed,* $\Phi = \Phi_1 \wedge \Phi_2$, *where* $\Phi_1 = ((a \leftrightarrow b) \wedge c) \vee (a \wedge \neg c \wedge d))$ *is a formula of* $PROP_{\{a,b,c,d\}}$ *and* $\Phi_2 = ((a \vee b) \leftrightarrow \neg e)$ *is a formula of* $PROP_{\{a,b,e\}}$.
A maximal sound policy for $\forall\{a,b\}\exists\{c,d\}\Phi_1$ *is given in Example 7. A maximal sound policy for* $\forall\{a,b\}\exists\{e\}\Phi_2$ *is the policy induced by the description* `Case` $a \vee b : \neg e$; $\neg(a \vee b) : e$ `End`.
Therefore, a maximal sound policy for P *is the policy induced by the following description:*
[`Case` $a \leftrightarrow b : (c,d)$; $a : (\neg c, d))$ `End`]
⊙ [`Case` $a \vee b : \neg e$; $\neg(a \vee b) : e$ `End`].

The compilation approach

The compilation approach consists in generating first a *compiled form* σ of Φ enabling polytime conditioning and model finding:

Proposition 9 *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 conditioning and polytime model finding.* σ *is a tractable representation of a maximal sound policy for* P.

Proof: Given an instantiation $\vec{x} \in 2^X$, any 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 enable polytime conditioning. 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 9.

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 (Schrag 1996; 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 $\text{QBF}_{2,\forall}$.

Related Work

As illustrated in Proposition 1, a topic very close to the notion of policy is the notion of *certificate*. In the case of QBF and under the usual assumption of complexity theory, there is no way to represent in polynomial space or to check in polynomial time a certificate of membership to the class of positive or negative instances of QBF. One of the practical consequence for QBF solver designers is the impossibility to easily validate the answers of their solver. In the last two QBF evaluations (Le Berre *et al.* 2004; Narizzano, Pulina, & Tacchella 2006), one third of the solvers submitted were found incorrect.

(Kleine Büning, Subramani, & Zhao 2003; Zhao & Buning 2005) 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 $\text{QBF}_{2,\forall}$) has a polysize solution policy; furthermore, they do not consider partial policies. This shows their approach mainly orthogonal to ours.

(Benedetti 2005a) suggests to represent certificates for positive instances of QBF using a forest of OBDD. According to the author, in practice, the cost of verifying that certificate is reasonable (both in terms of space and time). However, reconstructing the certificate from the solver's trace may overcome the time needed to solve the problem.

(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 solution policies using a circuit-based representation scheme. The complexity of determining whether a given QBF has a solution 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.

Conclusion

In this paper we provided theoretical ground for solving function problems associated with QBF, and algorithmic techniques for solving (and representing solutions for) formulae $\forall X\ \exists\ \Phi$ from $\text{QBF}_{2,\forall}$.

The specificities of our work are the following ones: define partial, but maximal 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 $\mathrm{QBF}_{2,\forall}$ and show how techniques such as decomposition and compilation can be fruitfully exploited for computing maximal sound policies.

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

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

Benedetti, M. 2005a. Extracting Certificates from Quantified Boolean Formulas. In *IJCAI'05*, 47–53.

Benedetti, M. 2005b. sKizzo: a Suite to Evaluate and Certify QBFs. In *CADE'05*.

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

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.

Coste-Marquis, S.; Le Berre, D.; Letombe, F.; and Marquis, P. 2005. Propositional Fragments for Knowledge Compilation and Quantified Boolean Formulae. In *AAAI'05*, 288–293.

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.

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

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

GhasemZadeh, M.; Klotz, V.; and Meinel, C. 2004. Zqsat: a qsat solver based on zero-suppressed binary decision diagrams. Technical report, University of Trier.

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

Karp, R., and Lipton, R. 1980. Some connections between non-uniform and uniform complexity classes. In *STOC'80*, 302–309.

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.

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

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

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).

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

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

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

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.

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

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

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

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

Zhao, X., and Buning, H. K. 2005. Model-equivalent reductions. In *SAT'05*, 355–370.