Extracting Certificates from Quantified Boolean Formulas *

Marco Benedetti
Istituto per la Ricerca Scientifica e Tecnologica (IRST)
Via Sommarive 18, 38055 Povo, Trento, Italy
benedetti@itc.it

Abstract

A *certificate of satisfiability* for a quantified boolean formula is a compact representation of one of its models which is used to provide solver-independent evidence of satisfiability. In addition, it can be inspected to gather explicit information about the semantics of the formula. Due to the intrinsic nature of quantified formulas, such certificates demand much care to be efficiently extracted, compactly represented, and easily queried. We show how to solve all these problems.

1 Introduction

The term "certificate" has a fairly general meaning, originating in language recognition and complexity theory. Once *verified*, a certificate proves that the string it refers to actually belongs to a language of interest. Applied to logic, the term denotes any means of providing evidence of (un)satisfiability for a given statement, other than a refutationally-compete deductive approach. In essence, we verify that a given logical formula belongs to the language of (un)satisfiable statements.

The most natural *certificate of satisfiability* (sat-certificate) for a formula is an explicit representation of some of its *models*. A formula is indeed satisfiable if and only if some model makes it evaluate to true. The validity of a certificate can be verified by whoever is knowledgeable about the *evaluation apparatus* of the logic (deductive capabilities are unnecessary), independently of how it was obtained.

In this paper, we focus on sat-certificates for *Quantified Boolean Formulas* (QBFs). Such certificates have never been proposed or used so far for a number of reasons. First, the intrinsic nature of a QBF confers a tree-shaped structure to its models, whose explicit representation may become unaffordable. Second, theoretical arguments exist that make it unlikely to find polynomial-time verification procedures (QBF satisfiability is PSPACE-complete [Stockmeyer and Meyer, 1973]). Finally, present QBF solvers find it either impractical or not straightforward to collect all the information needed to construct a model. As a consequence, QBF models feature no commonly accepted representation (if any at all), and all the current solvers return a little more than a sat/unsat answer.

Despite these issues, sat-certificates for QBFs are extremely desirable thank to their potential benefits on applications and solvers. For example, a certificate is a conclusive means to judge conflicting answers given by different solvers on the very same instance. Clearly, this event reveals no finer problem than a bug in the implementation, which we might think is not worth considering. This happens fairly often though, and as long as we treat solvers as black boxes, a proof-of-satisfiability approach is the only realistic way to tell the truth. We quote [Le Berre *et al.*, 2004]:

> *The question of how to check the answer of the QBF solvers in an effective way is still unanswered* [...] *the question of what is a good certificate of satisfiability/unsatisfiability* [...] *remains open. This point is not only an issue for the QBF evaluation, but also for the implementation:* [...] *we had soundness problems with 4 QBF solvers.* [...]

Yet, a certificate is much more than a way to ensure satisfiability: It can be *inspected* to gather semantics from the underlying formula. This is of paramount importance in applications, where certificates add valuable information to a mere sat/unsat answer. For example, a sat answer to the propositional (PROP) encoding of (the negation of) a desired property over a logic circuit means that the circuit is faulty w.r.t. that property. But, it takes a certificate to outline a definite scenario in which the fault shows up. As opposed to QBF certificates, such PROP certificates are easy to represent and verify, hence they have had a wide application.

The relevance of certificates enlarges with the scope of application of the underlying logic. In this respect, QBF is a notable case with plenty of applications. Every problem that can be stated as a two-player finite game can be modeled in QBF. An insightful example is obtained by considering the famous game "Connect-4". It is known that the player who moves first can always win. The rules of the game and the existence of a winning strategy can be encoded into a QBF instance [Gent and Rowley, 2003], expected to be sat. Which is the winning strategy? A certificate would disclose such information: The first player would prevail by just inspecting the certificate at each move, whatever the opponent does.

The interesting point here is that many real-world applications can be modeled as two-player games: Unbounded

*This work is supported by PAT (*Provincia Autonoma di Trento*, Italy), under grant n. 3248/2003.

model checking for finite-state systems [Rintanen, 2001] and conformant planning [Rintanen, 1999]—just to name two relevant examples—have handy QBF formulations.

In the rest of this paper, after a brief introduction to QBFs and their models (Section 2), we present a solver-independent representation for QBF sat-certificates (Section 3). As expected, we are able to describe how to verify them (Section 4) *before* the more complex task of their extraction is addressed (Section 5). We conclude by discussing the implementation of our approach and the future work (Section 6).

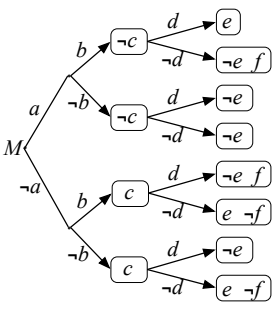
2 QBFs and their models

With no loss of generality, we consider QBFs in *prenex conjunctive normal form* (CNF). They consist in a *prefix* exhibiting an arbitrary number of alternations of existentially and universally quantified variables, followed by a *matrix*, i.e. a conjunction of clauses. For example:

$$\begin{array}{l}\forall a\forall b\exists c\forall d\exists e\exists f.(\neg b\vee e\vee f)\wedge(a\vee c\vee f)\wedge(a\vee d\vee e)\wedge\\ \quad(\neg a\vee\neg b\vee\neg d\vee e)\wedge(\neg a\vee b\vee\neg c)\wedge(\neg a\vee\neg c\vee\neg f)\wedge\\ \quad(a\vee\neg d\vee\neg e)\wedge(\neg a\vee d\vee\neg e)\wedge(a\vee\neg e\vee\neg f)\end{array}\quad(1)$$

Given a QBF F, we denote by $\widetilde{F}$ its matrix, by $var_\exists(F)$ ($var_\forall(F)$) the set of existentially (universally) quantified variables in F, and by $var_\forall(F,e)\subseteq var_\forall(F)$ the set of universal variables preceding (or *dominating*) $e\in var_\exists(F)$ in the prefix (we pose $\delta(e)\doteq|var_\forall(F,e)|$). Given a CNF matrix $\widetilde{F}$, the formula $\widetilde{F}*l$ is the CNF obtained by *assigning* the literal l, i.e. by removing from F each $\neg l$ literal and each clause containing l. This notation is readily extended to sets of literals. A matrix $\widetilde{F}$ is *satisfied* by a set of literals M (written $M\models\widetilde{F}$) when $\widetilde{F}*M$ is the empty formula.

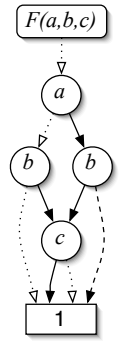
The alternations of quantifiers in the prefix guide us to extending this notion of satisfiability from matrixes to QBFs. For example, the satisfiability problem on (1) asks whether for each possible (consistent) combination of literals on a and b there exists a way to choose a literal on c such that for both possible literals over d, two literals on e and f exist such that the resulting set satisfies the matrix. Hence, a QBF model is a set of $|var_\exists(F)|$ functions, each one specifying the literal to be chosen (if any) on the existential variable e as a function of the choices on all the universal variables dominating e. To represent all these functions at once we can use a labeled tree, such as the one aside, depicting a model for (1). If we call *universal hypothesis* every consistent set of universal literals (or, equivalently, every assignment $[u_1=\psi_1,\ldots,u_n=\psi_n]$ to the universal variables, with $\psi_i\in\mathfrak{B}=\{0,1\}$), we may say, informally, that a tree structure like the one above is a model for a QBF with matrix $\widetilde{F}$ iff for every universal hypothesis U the set of existential literals collected along the branch individuated by U satisfies $\widetilde{F}*U$ (see [Büning and Zhao, 2004] for details).



3 Certificate representation

A QBF model can be represented *explicitly* by employing data structures such as trees or truth tables. Or, we may pursue *compactness* at the expense of managing an *implicit* representation[1] requiring computation to yield values.

An ideal certificate should be compact (easy to manage) *and* explicit (easy to verify and query). A successful tradeoff is obtained by employing *Binary Decision Diagrams* [Bryant, 1986]. We consider their *reduced ordered* version (ROBDDs, or just BDDs henceforth) with *complemented arcs*. A BDD $\mathcal{E}$ representing a total function $F(u_1,u_2,\ldots,u_n)$ from $\mathfrak{B}^n$ to $\mathfrak{B}$ is a directed acyclic graph with one root (labeled by F) and one sink node (labeled by "1"). Each internal node is labeled by one variable in $U=\{u_1,u_2,\ldots u_n\}$, and always has two children, one attached to the outcoming *then-arc*, the other to the *else-arc*. The *else-arc* may or may not be *complemented*. A unique path from the root to the sink is identified by assigning a value to each variable in U: The *then-arc* is chosen for variables assigned to 1, the *else-arc* is followed otherwise. The function F represented by $\mathcal{E}$ evaluates to 1 on $\langle\psi_1,\psi_2,\ldots,\psi_n\rangle\in\mathfrak{B}^n$ iff an even number of complemented arcs is encountered along the path defined by $\psi_1,\psi_2,\ldots,\psi_n$.



As an example, let us consider the BDD aside, where solid arrows denote then-arcs, while dashed (dotted) arcs are used for regular (complemented) else-arcs. It represents a binary function $F(a,b,c)$ of three binary variables a, b and c. It is, for example, $F(0,1,1)=1$ and $F(1,1,1)=0$. The represented function may be written as $F=b\wedge(a\vee c)\wedge(\neg a\vee\neg c)$. In a set-oriented interpretation, this BDD represents the *one-set* of F, i.e. the set having F as characteristic function. In our case, it stands for the set $\{\langle0,1,1\rangle,\langle1,1,0\rangle\}$ where F evaluates to 1.

The BDDs we utilize are *ordered* and *reduced*: The same variable ordering is followed along each path, and no two nodes representing the same set exist, so that each function has only one *canonic* representation. Furthermore, the version with complemented arcs is such that the set $\overline{S}$ is denoted by the same node as S (referred to with a complemented arc).

The BDD way of representing sets is regarded as *symbolic* in that it avoids the explicit enumeration of sets' elements in favor of a more abstract, diagram-based way of computing characteristic functions. Such representations may be exponentially more succinct than explicit ones (see [Wegener, 2000]), and all the operations on the sets/functions they represent (union/disjunction, intersection/conjunction, etc.) can be performed by manipulating the involved BDDs [Bryant, 1986]. With a small abuse of notation, we treat BDDs as if they were the sets they represent. For example, $x\in\mathcal{E}$ is an element in the subset of $\mathfrak{B}^n$ individuated by $\mathcal{E}$.

When we manage collections of BDDs, canonicity spans over their set of nodes as a whole. This allows the sharing of structural information among diagrams. A BDD in such a set

[1] In [Büning and Zhao, 2004] propositional formulas and QBFs with free variables are used. *Implicitness* is not an issue for the authors as they focus on characterizing classes of models/formulas.

of interconnected diagrams—called a *forest*—is identified by a (complemented) arc pointing to its root node.

Definition 3.1 (QBF sat-certificate, validity) *A sat-certificate for a QBF F with $var_\exists(F)=\{e_1,\ldots,e_m\}$, $var_\forall(F)=\{u_1,\ldots,u_n\}$, and $\delta_i=\delta(e_i)$ is a forest of BDDs containing two roots $\langle\mathcal{E}_i^+,\mathcal{E}_i^-\rangle$ for each i in $[1,m]$. Both $\mathcal{E}_i^+$ and $\mathcal{E}_i^-$ are defined over $var_\forall(F,e_i)=\{u_1,\ldots,u_{\delta_i}\}$. The certificate*

$$\mathcal{C}(F)=[\langle\mathcal{E}_1^+,\mathcal{E}_1^-\rangle,\langle\mathcal{E}_2^+,\mathcal{E}_2^-\rangle,\ldots,\langle\mathcal{E}_m^+,\mathcal{E}_m^-\rangle]$$

is consistent *when $\forall i\in[1,m]$ it is $\mathcal{E}_i^+\cap\mathcal{E}_i^-=\emptyset$. It is* valid *for F when for any $\langle\psi_1,\ldots,\psi_n\rangle\in\mathfrak{B}^n$ the formula $\widetilde{F}_{[u_1=\psi_1,\ldots,u_n=\psi_n]}$ is satisfied by $\{e_i=s^{(i)}(\psi_1,\ldots,\psi_{\delta_i}), i\in[1,m]\}$, where the functions $s^{(i)}:\mathfrak{B}^{\delta_i}\to\mathfrak{B}$ are defined as*

$$s^{(i)}(\psi_1,\ldots,\psi_{\delta_i})=\begin{cases}1 & \text{if } \langle\psi_1,\ldots,\psi_{\delta_i}\rangle\in\mathcal{E}_i^+\\ 0 & \text{if } \langle\psi_1,\ldots,\psi_{\delta_i}\rangle\in\mathcal{E}_i^-\\ undef. & \text{otherwise}\end{cases}$$

In essence, a sat-certificate is a compact but explicit representation of the dependencies that have to exist between existential (dependent) and universal (independent) variables in order to satisfy the matrix whichever the universal hypothesis.

Lemma 3.1 *If $\mathcal{C}(F)$ is valid for a QBF F, then F is satisfiable. Every satisfiable QBF has at least one valid certificate.*

Proof sketch. A QBF is satisfiable iff it has at least one model, i.e. iff we find at least one tree-like structure (like the one introduced in Section 2), such that for every assignment $U=[u_1=\psi_1,\ldots,u_n=\psi_n]$ to the universal variables the set of existential literals collected along the branch individuated by U satisfies $\widetilde{F}*U$. Given a consistent certificate C for F, we insert the literal e_i into the label of the node reached following the $u_1=\psi_1,\ldots,u_{\delta_i}=\psi_{\delta_i}$ path iff $\langle\psi_1,\ldots,\psi_{\delta_i}\rangle\in\mathcal{E}_i^+$ (and, dually, $\neg e_i$ appears in the label iff $\langle\psi_1,\ldots,\psi_{\delta_i}\rangle\in\mathcal{E}_i^-$). By construction, if the certificate is valid according to the notion of validity given in Definition 3.1, the tree-like structure obtained is a model. □

A valid sat-certificate for (1) is depicted in Figure 1.

4 Certificate verification

The first thing we wish to do with a consistent certificate $\mathcal{C}$ for F is to verify its validity. We check that by choosing the truth values of the existential variables according to what the certificate suggests, we always satisfy the matrix.

Function `checkValidity`(*QBF F, certificate $\mathcal{C}$*)

Let $var_\exists(F)$ be $\{e_1,\ldots,e_m\}$;
Let $var_\forall(F)$ be $\{u_1,\ldots,u_n\}$;
Let $\mathcal{C}$ be $[\langle\mathcal{E}_1^+,\mathcal{E}_1^-\rangle,\langle\mathcal{E}_2^+,\mathcal{E}_2^-\rangle\ldots,\langle\mathcal{E}_m^+,\mathcal{E}_m^-\rangle]$;
forall $\Gamma\in F$ **do**
  Let Γ be $\psi_1\otimes u_{i_1}\vee\cdots\vee\psi_h\otimes u_{i_h}\vee$
    $\phi_1\otimes e_{j_1}\vee\cdots\vee\phi_k\otimes e_{j_k}$;
  $\mathcal{E}\leftarrow(\cup_{\phi_i=0}\mathcal{E}_{j_i}^+)\cup(\cup_{\phi_i=1}\mathcal{E}_{j_i}^-)$;
  if $\overline{\mathcal{E}}_{[u_{i_1}=\overline{\psi_1},\ldots,u_{i_h}=\overline{\psi_h}]}\neq\emptyset$ **then return** FALSE;
return TRUE;

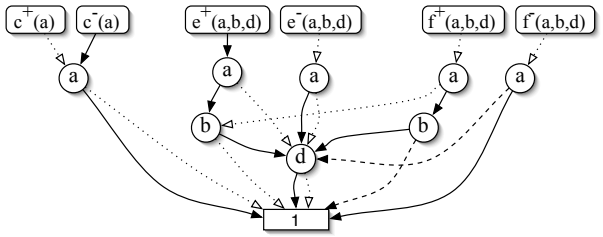


Figure 1: A BDD-based sat-certificate for the QBF (1).

An easy but impractical way of checking a certificate would be to check that $\mathcal{M}$ produces a satisfying assignment under all the possible universal hypotheses. Fortunately, the symbolic nature of the certificate helps us to perform a much more efficient, clause by clause, BDD-based verification. Let us use the exclusive or "$\otimes$" to construct literals out of variables ($\varphi\otimes v$ means v when $\varphi=0$, and $\neg v$ when $\varphi=1$).

Lemma 4.1 *The algorithm* `checkValidity` *answers* TRUE *on $\langle F,\mathcal{C}\rangle$ if and only if $\mathcal{C}$ is a valid certificate for F.*

Let us consider, for example, the clause $\neg u_1\vee e_1\vee u_2\vee\neg e_2\vee e_3$ under the prefix $\forall u_1\exists e_1\forall u_2\exists e_2\exists e_3$. The only relevant universal hypotheses for this clause are those assigning both $u_1{=}1$ and $u_2{=}0$: All the others immediately satisfy the clause via one of its universal literals. So, it remains to verify that under the assignment $[u_1{=}1,u_2{=}0]$ at least one of the three remaining literals in the clause is true, i.e. that every universal hypothesis containing $u_1=1$, $u_2=0$ falls within the one-set of at least one out of $\mathcal{E}_1^+$ (for e_1), $\mathcal{E}_2^-$ (for $\neg e_2$), and $\mathcal{E}_3^+$ (for e_3). This is a two-step check: First, we collect the universal hypotheses $\mathcal{E}=\mathcal{E}_1^+\cup\mathcal{E}_2^-\cup\mathcal{E}_3^+$ under which the clause is satisfied by some existential literal. Then, we check that all the hypotheses $\overline{\mathcal{E}}$ (in which no existential literal satisfies the clause) assign either $u_1=1$, or $u_2=0$, or both, so that the clause is satisfied by a universal literal.

The meaning of a successful verification is twofold: We are ensured that the formula is sat, and that the certificate encodes a model. Conversely, the verification fails when either the certificate is invalid or the formula is unsat (we cannot tell right away which circumstance has occurred). Validity check is a coNP-complete problem [Büning and Zhao, 2004].

5 Certificate Extraction

For certificate extraction to be symbolic in the same sense as our certificate is, we want it to work on a BDD-based representation of the problem. We describe one such representation (Section 5.1), and show how it relates to certificates (Section 5.2). Then, a two-step procedure is discussed to (a) evaluate the instance (Section 5.3), and (b) construct a certificate on the basis of the steps taken in the evaluation (Section 5.4).

5.1 Symbolic Formulas via Skolemization

The Skolem theorem shows how to transform any given *First Order Logic* (FOL) statement F into a *skolemized* formula $Sk(F)$ that has two properties: (1) $Sk(F)$ contains no existential quantifier, and (2) $Sk(F)$ is satisfiable iff F is satisfiable. Existential quantifiers are eliminated by replacing the

variables they bind with *Skolem functions* whose definition domains are appositely chosen to preserve satisfiability. In the *outer* form of skolemization, the function introduced for $e \in var_\exists(F)$ depends on the universal variables $var_\forall(F, e)$ that have e in their scope (for prenex formulas: all the universal variables to the left of e in the prefix).

Skolemization-based solvers replace the original formula F with the satisfiability-equivalent instance $Sk(F)$. Such instance is no longer propositional. Nevertheless, we are able to capture its semantics without exceeding the expressive power of propositional logic, by explicitly managing the truth values of the (interpretation of) skolem terms in each point of their definition domains, as shown in [Benedetti, 2005].

The duty we pay is a (possibly) exponential blowup in the size of the problem. BDDs come out to be precious in keeping this space explosion problem under control: What we actually manage is a *symbolic formula*, i.e. a compact BDD-based representation of a propositional instance representing the *definability* of the set of skolem terms introduced in $Sk(F)$.

Let as denote by $\Psi|_k$ the k-bit long prefix of $\Psi \in \mathfrak{B}^n, n \geq k$. The notion is extended to sets: $\mathcal{I}|_k = \{\Psi|_k . \Psi \in \mathcal{I}\}$.

Definition 5.1 (Symbolic formula) *A symbolic formula $\mathcal{F}$ is a BDD-based representation of a CNF instance. It consists of a* symbolic prefix $[e_1]_{\delta_1} \ldots [e_m]_{\delta_m}$ *on the variables* $var(\mathcal{F}) = \{e_1, \ldots, e_m\}$*, with* $0 \leq \delta_1 \leq \cdots \leq \delta_m$*, followed by a* symbolic matrix $\widetilde{\mathcal{F}}$*, i.e. a conjunction of* symbolic clauses*. A symbolic clause $\Gamma_\mathcal{I}$ is made up by a consistent set $\Gamma = [\varphi_1 \otimes e_{i_1}, \ldots, \varphi_h \otimes e_{i_h}]$ of literals on $var(\mathcal{F})$, and a (BDD represented) subset $\mathcal{I}$ of $\mathfrak{B}^{\delta(\Gamma)}$, $\delta(\Gamma) \doteq \max_{l \in \Gamma} \delta(l)$. The CNF represented by $\mathcal{F}$ is called* propositional expansion *of $\mathcal{F}$. It has variables $\{s^{(i)}_\Phi, i \in [1, m], \Phi \in \mathfrak{B}^{\delta(e_i)}\}$, and is defined as $Prop(\mathcal{F}) \doteq \wedge_{\Gamma_\mathcal{I} \in \mathcal{F}} Prop(\Gamma_\mathcal{I})$, where*

$$Prop(\Gamma_\mathcal{I}) \doteq \bigwedge_{\Psi \in \mathcal{I}} \varphi_1 \otimes s^{(1)}_{\Psi|_{\delta_1}} \vee \ldots \vee \varphi_m \otimes s^{(m)}_{\Psi|_{\delta_m}}$$

Symbolic formulas inherit the semantics of their propositional expansions (which we also call *ground* counterparts). Noticeably, a consistent set $\mathcal{M} = \{[l_1]_{\mathcal{I}_1}, \ldots, [l_m]_{\mathcal{I}_m}\}$ of symbolic literals satisfies $\mathcal{F}$ ($\mathcal{M} \models \mathcal{F}$) iff its expansion $Prop(\mathcal{M}) = \cup_{[l]_\mathcal{J} \in \mathcal{M}} Prop([l]_\mathcal{J})$ satisfies $Prop(\mathcal{F})$.

Definition 5.2 (Symbolic skolemization) *The symbolic skolemization $SymbSk(F)$ of a QBF F with $var_\exists(F) = \{e_1, \ldots, e_m\}$ is a symbolic formula with prefix $[e_1]_{\delta(e_i)} \cdots [e_m]_{\delta(e_m)}$, having one symbolic clause $[l_1, \ldots, l_h]_\mathcal{I}$ for each clause $\Lambda \in F$, where $\{l_1, \ldots, l_h\}$ are the existential literals in Λ, $\{\varphi_1 \otimes e_{i_1}, \ldots, \varphi_h \otimes e_{i_h}\}$ are the universal literals in Λ, and $\mathcal{I} = \{\langle \psi_i, \ldots, \psi_k \rangle \in \mathfrak{B}^k \mid \forall j . \psi_{i_j} \neq \varphi_j\}$, $k = \delta(\Lambda)$.*

As an example, the symbolic skolemization of (1) is given in Figure 2. In essence, a symbolic skolemization $\mathcal{F} = SymbSk(F)$ is a compact representations for a purely existential instance $Prop(\mathcal{F})$ having the following key property.

Theorem 5.1 *For any QBF F, $Prop(SymbSk(F)) \stackrel{sat}{\equiv} F$.*

Proof sketch. By applying outer skolemization as described in [Benedetti, 2005] we turn the QBF instance F into a sat-equivalent purely universal formula: We substitute every existential variable v dominated by $\{u_1, u_2, \ldots, u_n\} \subseteq var_\forall(F)$ with a *Skolem function* $s^v(u_1, u_2, \ldots, u_n)$. For example, the instance (1) with matrix $\widetilde{N}$, is sat-equivalent to

$$\forall a \forall b \forall d .\ \widetilde{N}[s^c(a, b)/c,\ s^e(a, b, d)/e,\ s^f(a, b, d)/f] \quad (2)$$

Then, we propositionally encode the *definability* of the Skolem terms introduced, leveraging a noteworthy feature: They all map $\mathfrak{B}^n$ onto $\mathfrak{B}$ (for some $n \geq 0$), hence they are fully specified by 2^n boolean parameters, and have a direct CNF representation. For example, if we denote by $\{s^c_{\alpha\beta}, \langle \alpha, \beta \rangle \in \mathfrak{B}^2\}$ the four boolean parameters representing the truth value of s^c over $\langle x, y \rangle$, we obtain for $s^c(a, b)$ the following *propositional skolemization* $Sk(s^c) \equiv s^c(a, b)$:

$$(a \vee b \vee s^c_{00}) \wedge (a \vee \neg b \vee s^c_{01}) \wedge (\neg a \vee b \vee s^c_{10}) \wedge (\neg a \vee \neg b \vee s^c_{11})$$

By replacing each $v \in var_\exists(F)$ with $Sk(s^v)$ we obtain $Sk(F)$

$$\exists \underline{s}^c \exists \underline{s}^e \exists \underline{s}^f \forall a \forall b \forall d .\ \widetilde{N}[Sk(s^c)/c,\ Sk(s^d),\ Sk(s^e)/e]$$

where $\exists \underline{s}^c$ stands for $\exists s^c_{00} \exists s^c_{01} \exists s^c_{10} \exists s^c_{11}$, and similarly for the other functions. This formula is easily turned—clause by clause—into a CNF. By distributing the connectives, removing clauses with complementary literals, and eliminating literals over universal quantifiers, we obtain $2^{\delta(\Gamma)-m}$ clauses out of a QBF clause Γ with m universal literals. For example, we obtain 2^{3-1}=4 clauses from the clause $a \vee c \vee f$ in (1):

$$(s^c_{00} \vee s^f_{000}) \wedge (s^c_{00} \vee s^f_{001}) \wedge (s^c_{01} \vee s^f_{010}) \wedge (s^c_{01} \vee s^f_{011})$$

All the clauses coming out of a given QBF clause mention the same skolem functions. What makes one differ from another are the subscripted indexes. This allows us to write them compactly by representing function names apart from indexes. For example, the four clauses above may be succinctly represented as $\Gamma_\mathcal{I} = [c, f]_{\{000,001,010,011\}}$, where $\mathcal{I} \subseteq \mathfrak{B}^{\delta(\Gamma)}$ contains one element per clause, and the i-th component of each $\Psi \in \mathcal{I}$ refers to the universal variable u_i: Once $\Psi \in \mathcal{I}$ is selected, each literal l obtains its own subscript by *projecting* Ψ onto the subspace related to the first $\delta(l)$ components, written $\Psi|_{\delta(l)}$. For example, given Ψ=010 it is $\Psi|_{\delta(c)} = \Psi|_2$=01 and $\Psi|_{\delta(f)} = \Psi|_3$=010, hence $[c, f]_{\{010\}} = s^c_{01} \vee s^f_{010}$. This property allows us to recover the ground meaning of a factored clause through the $Prop$ function. The factored representation becomes *symbolic* as soon as we represent and manipulate sets of indexes via BDDs over $var_\forall(F)$, thus obtaining the (linear-size) representation $SymbSk(F)$. □

For example, the propositional expansion of the formula in Figure 2 yields a CNF instance equivalent to (1).

5.2 From symbolic models to certificates

The connection between a symbolic model and a certificate is so close that the former is smoothly turned into the latter. Let $\mathfrak{i}(\mathcal{M}, l) \doteq \cup_{[l]_\mathcal{I} \in \mathcal{M}} \mathcal{I}$ be the indexes on the literal l mentioned in a symbolic model $\mathcal{M}$.

Theorem 5.2 *If $\mathcal{M} \models SymbSk(F)$, then the certificate*

$$\mathcal{C}(\mathcal{M}) \doteq [\langle \mathcal{C}^+_{e_1}, \mathcal{C}^-_{e_1} \rangle, \langle \mathcal{C}^+_{e_2}, \mathcal{C}^-_{e_2} \rangle, \ldots, \langle \mathcal{C}^+_{e_m}, \mathcal{C}^-_{e_m} \rangle]$$

with $\mathcal{C}^+_e = \mathfrak{i}(\mathcal{M}, e)$, and $\mathcal{C}^-_e = \mathfrak{i}(\mathcal{M}, \neg e)$ is valid for F.

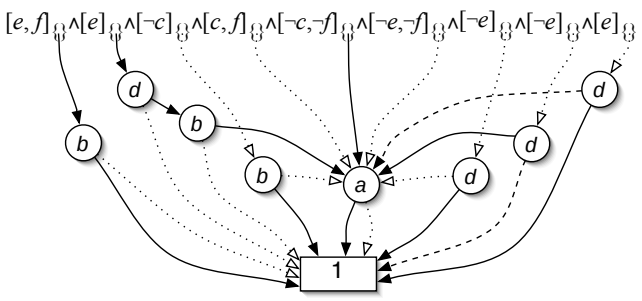


Figure 2: Symbolic propositional skolemization for (1).

Proof. The propositional variable s^i_Ψ in $Prop(\mathcal{F})$, $\mathcal{F} = SymbSk(F)$ represents, for each $i \in [1,m]$ and each $\Psi = \langle \psi_1, \ldots, \psi_k \rangle \in \mathfrak{B}^k$, $k{=}\delta(e_i)$, the value the Skolem function $s^{(i)}(u_1, \ldots, u_k)$ (introduced to skolemize $e_i \in var_\exists(F)$) assumes under the universal hypothesis $[u_1{=}\psi_1, \ldots, u_k{=}\psi_k]$. By construction, $\mathcal{F}$ embodies the definability of the skolem terms, i.e. all the mutual constraints among (interpretation of) terms that have to be obeyed to always satisfy the matrix. A model $\mathcal{M}$ for $\mathcal{F}$ is a way to comply with all these constraints at once: $s^{(i)}(\psi_1, \ldots, \psi_k)$ has to evaluate to 1 if $s^{(i)}_\Psi \in Prop(\mathcal{M})$, to 0 if $\neg s^{(i)}_\Psi \in Prop(\mathcal{M})$, and is unconstrained otherwise. By comparing the resulting functions

$$s^e(\psi_1, \ldots, \psi_k) = \begin{cases} 1 & \text{if } \langle \psi_1, \ldots, \psi_k \rangle \in \mathcal{C}^+_e \\ 0 & \text{if } \langle \psi_1, \ldots, \psi_k \rangle \in \mathcal{C}^-_e \\ undefined & \text{otherwise} \end{cases}$$

with Definition 3.1 (valid certificates) the thesis follows. □

5.3 Evaluating symbolic formulas

We devote our attention to *solution-based* procedures [Biere, 2004; Pan and Vardi, 2004; Benedetti, 2005], as they may be lifted to extract models *symbolically*. We consider a solver that manipulates symbolic clauses through rules designed to achieve at once on $\Gamma_\mathcal{I}$ the same result ground rules would obtain if applied separately to each clause in $Prop(\Gamma_\mathcal{I})$.

- *Resolution*: Given two *resolving* clauses containing the same variable e in opposite polarities, we construct a necessary consequence—called *resolvent* clause—made up by all the literals from both originating clauses apart from e and $\neg e$. For the symbolic version to produce at most one symbolic resolvent per step, we only resolve on variables with the same universal depth as all the clauses in which they appear. The resolvent of $\Gamma_\mathcal{I}$ and $\Gamma'_{\mathcal{I}'}$ on e, with $e{\in}\Gamma$, $\neg e{\in}\Gamma'$, $\delta(\Gamma){=}\delta(\Gamma'){=}\delta(e)$, is[2].
$$(\Gamma \setminus \{e\} \cup \Gamma' \setminus \{\neg e\})_{\mathcal{I}\cap\mathcal{I}'} \tag{3}$$
If either $\mathcal{I} \cap \mathcal{I}' = \emptyset$ or Γ and Γ' share further couples of complementary literals other than e and $\neg e$, the resolvent represents an empty set of clauses and is removed.
- *Subsumption*: $\Gamma_\mathcal{I}$ is subsumed by $\Gamma'_{\mathcal{I}'}$ when all the ground clauses in $\Gamma_\mathcal{I}$ are subsumed by some clause in $\Gamma'_{\mathcal{I}'}$, i.e. when $\Gamma' \subseteq \Gamma$ and $\mathcal{I} \subseteq \mathcal{I}'$. Under this condition, $\Gamma_\mathcal{I}$ can be removed. In *partial* subsumption ($\Gamma' \subseteq \Gamma$ but $\mathcal{I} \not\subseteq \mathcal{I}'$) the subsumed clause is replaced by $\Gamma_{\mathcal{I}\cap\overline{\mathcal{I}}'}$.
- *Substitution*: We write $[l/a]_\mathcal{J}$, $\delta(l) \leq \delta(a)$, to denote the substitution of $l_{\Psi|_{\delta(\phi)}}$ for a_Ψ for each $\Psi \in \mathcal{J}$:
$$\Gamma_\mathcal{I}[l/a]_\mathcal{J} = (\Gamma[l/a])_{\mathcal{I}\cap\mathcal{J}} \wedge \Gamma_{\mathcal{I}\cap\overline{\mathcal{J}}} \tag{4}$$

The rules above establish primitive manipulation capabilities for the solver. It arranges these components into the following higher-level, satisfiability-preserving transformations.

Assignment. When $[l]_\mathcal{J}$ is realized to be a consequence of $\mathcal{F}$ (i.e.: for all $\Psi \in \mathcal{J}$, it is $Prop(\mathcal{F}) \vdash l_\Psi$), the formula is simplified to $\mathcal{F} * [l]_\mathcal{J}$ by *assigning* $[l]_\mathcal{J}$, i.e. resolving against $[\neg l]_\mathcal{J}$ and subsuming against $[l]_\mathcal{J}$.

Equivalence reasoning. If the solver discovers a proof that $[a] \overset{\mathcal{J}}{\leftrightarrow} [l]$ (i.e.: $\forall \Psi{\in}\mathcal{J}$, $Prop(\mathcal{F}) \vdash a_{\Psi|_{\delta(a)}} {\leftrightarrow} l_\Psi$), it is entitled to simplify $\mathcal{F}$ by applying the substitution $[l/a]_\mathcal{J}$.

Redundancy removal. Subsumption is applied to eliminate subsumed clauses. This may heavily reduce the burden on the solver, while logical equivalence is preserved.

Variable elimination. The clauses containing a variable d are replaced by the set of resolvents of every clause containing d against each clause containing $\neg d$ (Figure 3). Symbolic elimination rules out one symbolic variable, hence all the related ground variables at once.

As far as we are concerned, a solution-based evaluation procedure is an algorithm that produces a (memory and time affordable) sequence of instantiations of the above rules, and guarantees to end up with the empty formula on sat instances, and with a contradiction (an empty clause) on unsat formulas. Each step is a back-chaining reduction from a problem $\mathcal{F}'$ to a problem $\mathcal{F}$. Hence, a model for the original instance is *not* directly extracted. Rather, satisfiability equivalence guarantees that at each step, should $\mathcal{F}$ have a model, than a model for $\mathcal{F}'$ could be (easily) derived.

This apparent drawback suddenly turns into an advantage. It indeed allows to *decouple* evaluation from model reconstruction, with almost no overhead for the former and a clear semantics for the latter. The two meshes of the chain are connected through an *inference log*, produced by the solver, and subsequently read by a *model reconstructor*.

Definition 5.3 (Inference log and trace) *An* inference log *is a list of entries, each one describing an instantiation of one of the above* sat*-preserving transformations. It contains:*

- $\langle \texttt{assign}, l, \mathcal{J} \rangle$*, for an assignment* $[l]_\mathcal{J}$.
- $\langle \texttt{subst}, a, l, \mathcal{J}, \mathcal{G} \rangle$*, for a substitution* $[l/a]_\mathcal{J}$*, where* $\mathcal{G}$ *is the set of clauses containing* $\{a, \neg l\}$ *or* $\{\neg a, l\}$.
- $\langle \texttt{elim}, d, \mathcal{G}^+, \mathcal{G}^- \rangle$*, for the elimination of* d*, where* $\mathcal{G}^+$ *and* $\mathcal{G}^-$ *are the sets of clauses containing* d *and* $\neg d$.

Let us denote with $\mathcal{I}(\mathcal{F}, \mathbf{op})$ *the formula obtained by applying to* $\mathcal{F}$ *the transformation described by the entry* $\mathbf{op}$*. An inference log* $\mathcal{L} = [\mathbf{op}_1, \mathbf{op}_2, \ldots, \mathbf{op}_t]$ *induces an* inference trace $[\mathcal{F}_0, \mathcal{F}_1, \ldots, \mathcal{F}_t]$ *where* $F_i = \mathcal{I}(\mathcal{F}_{i-1}, \mathbf{op}_i)$*, with* $\mathcal{F}_0 = \mathcal{F}$*. A log such that* $\mathcal{F}_t$ *is empty is called* sat-log *for* $\mathcal{F}$.

Clearly, a sat-log for $\mathcal{F}$ exists iff $\mathcal{F}$ is sat. For example, the reader may verify that Figure (4) depicts a sat-log for (1).

[2] To simplify the notation, we omit the projection of the index sets of derived clauses onto the proper subspaces: Whenever $\Gamma_\mathcal{I}$ is derived, the clause actually inserted into the formula is $\Gamma_{\mathcal{I}|_{\delta(\Gamma)}}$.

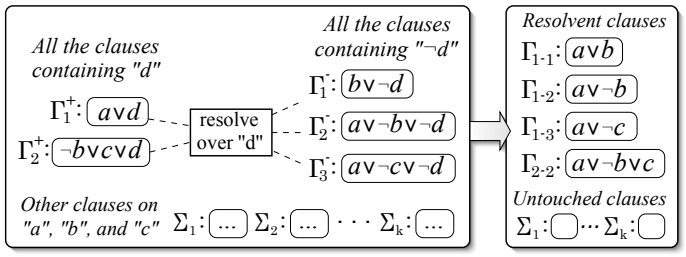


Figure 3: Moving from F' to F by eliminating the variable d.

#1	$\langle \texttt{substitute}, f, \neg e, \{010, 011\}, \{[e,f]_{\{010,011,110,111\}}, [\neg e, \neg f]_{\{000,001,010,011\}}\}\rangle$
#2	$\langle \texttt{assign}, \neg e, \{001, 011, 100, 110\}\rangle$
#3	$\langle \texttt{resolve}, f, \{[f]_{\{110\}}, [c,f]_{\{000,001\}}, [e,f]_{\{111\}}\}, \{[\neg c, \neg f]_{\{100,101,110,111\}}, [\neg e, \neg f]_{\{000\}}\}\rangle$
#4	$\langle \texttt{assign}, e, \{111\}\rangle$
#5	$\langle \texttt{resolve}, c, \{[c]_{\{00,01\}}\}, \{[\neg c]_{\{10,11\}}\}\rangle$

Figure 4: A sat-log that solves the QBF (1).

5.4 Inductive model reconstruction

Once a sat-log is known, the reconstructor comes into play. It trusts the solver about the log being a sat-log, and parses it *backward*, reasoning by induction on the number of entries:

Base case. At the end of the inference trace we find the empty formula $\mathcal{F}_t$, satisfied by an empty model $\mathcal{M}_t$.

Inductive case. Given a model $\mathcal{M}_i$ for $\mathcal{F}_i$, the reconstructor computes a model $\mathcal{M}_{i-1} = \mathcal{R}(\mathcal{M}_i, \mathbf{op}_i)$ for $\mathcal{F}_{i-1}$ by reasoning on how $\mathbf{op}_i$ turned $\mathcal{F}_{i-1}$ into $\mathcal{F}_i$.

This leads to a model $\mathcal{M}_0$ for $\mathcal{F}$, hence to a certificate for F, once the function $\mathcal{R}$ has been properly defined. Let us denote by $\mathtt{i}(\mathcal{F}) \doteq \cup_{\Gamma_\mathcal{I} \in \mathcal{F}} \mathcal{I}$ the set of indexes mentioned in $\mathcal{F}$.

Theorem 5.3 *When applied to any sat-log for $\mathcal{F}$, the inductive model reconstruction procedure defined by the following function $\mathcal{R}$ computes a model $\mathcal{M}_0$ for $\mathcal{F}_0 = \mathcal{F}$.*

$\mathbf{op}$	$\mathcal{R}(\mathcal{M}, \mathbf{op})$
$\langle \texttt{assign}, l, \mathcal{I}\rangle$	$\mathcal{M} \cup \{[l]_\mathcal{I}\}$
$\langle \texttt{subst}, e, l, \mathcal{I}, \mathcal{G}\rangle$	$\mathit{ext}^{\mathsf{s}}(\mathcal{M} \cup \{[e]_{\mathcal{I} \cap \mathtt{i}(\mathcal{M},l)}, [\neg e]_{\mathcal{I} \cap \mathtt{i}(\mathcal{M},\neg l)}\})$
$\langle \texttt{elim}, e, \mathcal{G}^+, \mathcal{G}^-\rangle$	$\mathcal{M}^{\mathsf{e}} \cup \{[e]_{\mathtt{i}(\mathcal{G}^+ * \mathcal{M}^{\mathsf{e}})}, [\neg e]_{\mathtt{i}(\mathcal{G}^- * \mathcal{M}^{\mathsf{e}})}\}$

where $\mathit{ext}^{\mathsf{s}}(\mathcal{M}) \doteq \mathcal{M} \cup [e]_{\mathtt{i}(\mathcal{G} * \mathcal{M})}$, *and* $\mathcal{M}^{\mathsf{e}} \doteq \mathit{ext}^{\mathsf{e}}(\mathcal{M})$, *with*

$$\mathit{ext}^{\mathsf{e}}(\mathcal{M}) = \begin{cases} \mathit{ext}(\mathcal{M} \cup [v]_\mathcal{I}) \text{ if } \exists \langle \Gamma_\mathcal{I}, \Gamma'_{\mathcal{I}'}\rangle \in \mathcal{G}^+ * \mathcal{M} \times \mathcal{G}^- * \mathcal{M} \\ \qquad \text{with } e \neq v, v \in \Gamma \cap \overline{\Gamma'},\ \mathcal{I} \cap \mathcal{I}' \neq \emptyset \\ \mathcal{M}, \text{ otherwise} \end{cases}$$

Proof. For each $\mathbf{op}$ we show that $\mathcal{R}(\mathcal{M}, \mathbf{op}) \models \mathcal{F}'$, working under the inductive hypothesis of knowing a model $\mathcal{M}$ for $\mathcal{F} = \mathcal{I}(\mathcal{F}', \mathbf{op})$ (i.e. $\mathcal{F} * \mathcal{M}$ is the empty formula).

Assignment. $\mathcal{F} * \mathcal{M} = (\mathcal{F}' * [l]_\mathcal{I}) * \mathcal{M} = \mathcal{F}' * (\{[l]_\mathcal{I}\} \cup \mathcal{M})$ is empty, hence $\{[l]_\mathcal{I}\} \cup \mathcal{M}$ satisfies $\mathcal{F}'$.

Equivalence. Let $\mathcal{F}'$ be $\mathcal{F}'_0 \cup \mathcal{F}'_1 \cup \mathcal{F}'_{2+} \cup \mathcal{F}'_{2-}$, where no clause in $\mathcal{F}'_0$ mentions e, all the clauses in $\mathcal{F}'_1$ mention e but not l, $\neg l$; all the clauses in $\mathcal{F}'_{2+}$ contain both e and l (or $\neg e$ and $\neg l$), and the clauses in $\mathcal{F}'_{2-}$ contain e and $\neg l$ (or $\neg e$ and l). $\mathcal{M}$ satisfies $\mathcal{F} = \mathcal{F}'[l/e]_\mathcal{I} = \mathcal{F}_0 \cup (\mathcal{F}'_1 \cup \mathcal{F}'_{2+})[l/e]_\mathcal{I}$, hence it satisfies $\mathcal{F}'_0 = \mathcal{F}_0$ with no modification. It also satisfies $\mathcal{F}'_1 \cup \mathcal{F}'_{2+}$ provided we mirror on e the assignments collected so far over l, by adding $[e]_{\mathcal{I} \cap \mathtt{i}(\mathcal{M},l)}$ and $[\neg e]_{\mathcal{I} \cap \mathtt{i}(\mathcal{M},\neg l)}$. In general, the resulting model $\mathcal{M}^+ = \mathcal{M} \cup \{[e]_{\mathcal{I} \cap \mathtt{i}(\mathcal{M},l)}, [\neg e]_{\mathcal{I} \cap \mathtt{i}(\mathcal{M},\neg l)}\}$ satisfies all the clauses in $\mathcal{F}'$ but some in $\mathcal{F}'_{2-}$: By construction, such clauses failed to pass to $\mathcal{F}$ because $\mathcal{F}'_{2-}[l/e]_\mathcal{I} = \mathcal{G}[l/e]_\mathcal{I}$ is empty (clauses satisfied by complementary $e/\neg l$ literals). To satisfy the (possibly) nonempty remaining set $\mathcal{G} * \mathcal{M}^+$ of clauses, we extend $\mathcal{M}^+$ by applying the $\mathit{ext}^{\mathsf{s}}$ function: It adds an arbitrary truth assignment to all (and only) the indexes for e mentioned in $\mathcal{G} * \mathcal{M}^+$, i.e. it adds $[e]_{\mathtt{i}(\mathcal{G} * \mathcal{M}^+)}$.

Variable elimination. We focus on the insightful ground-case proof (see Figure 3). The extension to the symbolic case is a matter of notation. Suppose that the model M for F satisfies at least one clause of each $\langle \Gamma_i^+, \Gamma_j^-\rangle$ couple, $i \in [1,n], j \in [1,m]$. This implies that it either satisfies the whole set $G_d^+ = \wedge_i \Gamma_i^+$, or the whole $G_d^- = \wedge_i \Gamma_i^-$, or both. Hence, we are free to choose a literal on d in such a way to satisfy F': $M \cup \{a\}$ satisfies F' when $G_d^+ * M$ is non-empty, $M \cup \{\neg a\}$ when $G_d^- * M$ is non-empty, while M itself suffices when both sets are empty. Now, we show that every model M for F either satisfies at least one clause in each $\langle \Gamma_i^+, \Gamma_j^-\rangle$ couple, or can be extended to a model $\mathit{ext}^{\mathsf{e}}(M)$ that fulfills this property. The former case happens (at least) when no resolvent clause has been satisfied by complementary literals during variable elimination. In this case, F contains exactly $m * n$ resolvents, one for each couple in $G_d^+ \times G_d^-$. Every model of a resolvent is valid for at least one of its resolving clauses, hence the thesis. Now, suppose that the couple $\langle \Gamma_i^+, \Gamma_j^-\rangle$ is not satisfied by M. It follows that M doesn't model the resolvent of Γ_i^+ and Γ_j^-, hence such resolvent failed to pass to F: It was satisfied by complementary literals on some still unassigned variable $v \neq d$. A literal on v can be arbitrarily added to the model under construction to satisfy either Γ_i^+ or Γ_j^- (no conflict arises: v would have not appeared as unassigned if involved in any past—w.r.t. the solver's standpoint—inference step). This *extension* of M to $\mathit{ext}^{\mathsf{e}}(M)$ is repeated until no unsatisfied couple is left. □

Examples: $M' = \{a, \neg b, c\}$ is a model for $F' = (\neg a \vee c) \wedge (a \vee b \vee \neg c) \wedge (\neg a \vee \neg b)$ because $M = \{a, \neg b\}$ is a model for $F = F' * c = (a \vee b) \wedge (\neg a \vee \neg b)$. $M' = \{\neg a, b, c\}$ is a model for $F' = (\neg a \vee c) \wedge (a \vee b \vee \neg c) \wedge (\neg a \vee \neg b)$ because $M = \{b, c\}$ is a model for $F'[\neg c/a] = c \wedge (b \vee \neg c) \wedge (c \vee \neg b)$. In Figure 3, the assignment $M_1 = \{a, b, c\}$ satisfies F and also F': d is left unassigned. The model $M_2 = \{a, \neg b\}$ satisfies G_d^+ but not G_d^-, hence $M' = M_2 \cup \{\neg d\}$ is constructed to satisfy F'. The couple $\langle \Gamma_2^+, \Gamma_1^-\rangle$ is untouched by $M_3 = \{a\}$, so no truth value for d helps. But the resolvent of Γ_2^+ and Γ_1^- was satisfied by complementary literals on b, so we construct $\mathit{ext}^{\mathsf{e}}(M_3) = M_3 \cup \{b\}$, then satisfy F' with $M' = \mathit{ext}^{\mathsf{e}}(M_3) \cup \{\neg d\} = \{a, b, \neg d\}$. As a complete example, the reader may verify that by performing inductive model reconstruction according to the rules given in Theorem 5.3 over the sat-log in Figure 4 for the $SymbSk(F)$ in Figure 2, a model is extracted which through Theorem 5.2 produces the sat-certificate in Figure 1.

6 Discussion and Conclusions

We presented a solution to an open question on QBFs, namely the problem of representing, verifying and extracting their sat-certificates. The importance of such solution is twofold: It gives means of conveying solver-independent evidence of satisfiability, and enables the extraction of precious information from certified formulas. The former feature can be used, for example, to effectively test the decisions of QBF solvers. The latter contribution is valuable to applications, in that a certificate is needed to exemplify a definite scenario in which QBF-encoded problems reveal their satisfiability. For example, a sat-answer suffices to know that at least one winning strategy exists in a QBF-encoded two-player game, but it takes a certificate to exhibit an actual strategy.

Stand-alone certificates convey no self-contained semantics, as the meaning of each variable is a piece of information held by the "encoders". To allow semantics' owners to interrogate their own QBF models, we have implemented a solver/verifier suite [Benedetti, 2004b; 2004a] to produce, verify, dump to file (in open formats), and query certificates.

This implementation is helping to shed light on further open questions on QBFs. For example: *Is QBF certification impractical*? Let us define the certification of F "impractical" when it is affordable to solve F while it is not feasible to memorize and/or verify any certificate $\mathcal{C}(F)$. Our approach suggests that impracticality is not an issue. For instance, the certificates in Table 1 are well within the manipulation capabilities of current machines, while the formulas they certify are hard for present solvers[3]. Another crucial question we are enabled to address is the following: *Given a certificate $\mathcal{C}$ for F, how much in practice the trio $\langle F, \mathcal{C}, \text{verify}\rangle$ improves on the couple $\langle F, \text{decide}\rangle$ as a means to prove that F is* sat? Metrics such as the $time \cdot space$ product will be used to precisely compare the two strategies. Our results already suggest that the improvement is quite large, as exemplified in Table 1.

We observed a surprising phenomenon: The time taken to reconstruct a model may overcome the time needed to solve the instance. This is rather unusual when compared, for example, with search-based SAT reasoning, where a model is extracted with no overhead on the satisfiability decision. Conversely, we have not yet observed an expected phenomenon: `checkValidity` would operate in polynomial time should we employ a constant-time BDD oracle. The non-polynomiality of verification stems from the size of the forest of BDDs, which should grow exponentially for some parametrically scalable family of instances. This unfavorable phenomenon *doesn't* show up in Table 1: Certificates scale up polynomially with instance size. These effects and upper bounds on the certificate size will be further investigated.

We showed how to extract certificates using a particular class of QBF solvers (the skolemization-based ones). However, the BDD-based representation we employ is solver-independent. It comes out that not only the mere representation but also the technique for constructing certificates can be lifted to work with other families of QBF-solvers. The key ingredient stays the same: Inductive model reconstruction detached from instance evaluation, with an inference log in between. What changes is the kind of information recorded in the log (and the way it is to be interpreted).

We are extending our technique towards solvers based on (1) non-symbolic q-resolution, (2) SAT reasoning, and (3) symbolic/non-symbolic DPLL-like branching reasoning. The ultimate goal is to build a QBF model reconstructor able to extract certificates by interpreting *generic* QBF inference logs[4], whichever the evaluation strategy adopted to solve the instance (including hybrid strategies such as the one leveraged by the QBF solver sKizzo [Benedetti, 2004b]).

instance	$\forall$	$\exists$	T_s	T_r	T_v	$\lvert\mathcal{L}\rvert$	$\lvert\mathcal{C}\rvert$
adder-2	27	37	0.1	0.1	0.1	38	$3.7 \cdot 10^2$
adder-4	106	174	0.2	0.1	0.1	165	$7.1 \cdot 10^3$
adder-6	237	411	0.6	2.3	0.1	384	$5.6 \cdot 10^4$
adder-8	420	748	4.6	35.6	0.4	695	$1.1 \cdot 10^5$
adder-10	755	1185	40.2	537.2	4.6	1098	$4.1 \cdot 10^6$

Table 1: A family of QBF encodings with $\forall\exists\forall\exists$ alternation. We report: the number of existential ($\exists$) and universal ($\forall$) variables, the time taken to solve/reconstruct/verify (T_s,T_r,T_v), the size of the log ($|\mathcal{L}|$, number of steps) and of the certificate ($|\mathcal{C}|$, number of nodes).

Acknowledgments

We thank Amedeo Cesta and Gigina Aiello for their comments on how to improve the organization of the paper, Marco Cadoli for helpful discussions on technical issues, and the anonymous referees for their precious remarks.

References

[Benedetti, 2004a] M. Benedetti. sKizzo: a QBF Decision Procedure based on Propositional Skolemization and Symbolic Reasoning. Technical Report 04-11-03, ITC-irst, 2004.

[Benedetti, 2004b] M. Benedetti. `sKizzo`'s web site, `sra.itc.it/people/benedetti/sKizzo`. 2004.

[Benedetti, 2005] M. Benedetti. Evaluating QBFs via Symbolic Skolemization. In *Proc. of the 11th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR04)*, number 3452 in LNCS. Springer, 2005.

[Biere, 2004] A. Biere. Resolve and Expand. In *Proc. of SAT'04*, 2004.

[Bryant, 1986] R. E. Bryant. Graph-based algorithms for Boolean function manipulation. *IEEE Transaction on Computing*, C-35(8):677–691, 1986.

[Büning and Zhao, 2004] H. K. Büning and X. Zhao. On Models for Quantified Boolean Formulas. In *Proc. of SAT'04*, 2004.

[Gent and Rowley, 2003] I. Gent and A. Rowley. Encoding Connect-4 using Quantified Boolean Formulae. Technical Report 68-2003, APES Research Group, 2003.

[Le Berre *et al.*, 2004] D. Le Berre, M. Narizzano, L. Simon, and A. Tacchella. Second QBF solvers evaluation, avaliable on-line at `www.qbflib.org`. 2004.

[Pan and Vardi, 2004] G. Pan and M.Y. Vardi. Symbolic Decision Procedures for QBF. In *Proc. of the 10^{th} Conf. on Princip. and Practice of Constraint Programming*, 2004.

[Rintanen, 1999] J. Rintanen. Construction Conditional Plans by a Theorem-prover. *Journal of A. I. Research*, pages 323–352, 1999.

[Rintanen, 2001] J. Rintanen. Partial implicit unfolding in the davis-putnam procedure for quantified boolean formulae. In *Proceedings of the International Conference on Logic for Programming, Artificial Intelligence and Reasoning (LPAR'01)*, 2001.

[Stockmeyer and Meyer, 1973] L. J. Stockmeyer and A. R. Meyer. Word Problems Requiring Exponential Time. In *In 5th Annual ACM Symposium on the Theory of Computing*, 1973.

[Wegener, 2000] Ingo Wegener. *Branching Programs and Binary Decision Diagrams*. Monographs on Discrete Mathematics and Applications. SIAM, 2000.

[3] The "adder" family belongs to a set of benchmarks related to equivalence checking of partial implementations of circuits, and is regarded as extremely challenging [Le Berre *et al.*, 2004]

[4] A standard log format for each inference rule is required.