Evaluating QBFs via Symbolic Skolemization

Marco Benedetti[§]

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

Abstract. We describe a novel decision procedure for Quantified Boolean Formulas (QBFs) which aims to unleash the hidden potential of quantified reasoning in applications. The Skolem theorem acts like a glue holding several ingredients together: BDD-based representations for boolean functions, search-based QBF decision procedure, and compilation-to-SAT techniques, among the others. To leverage all these techniques at once we show how to evaluate QBFs by symbolically reasoning on a compact representation for the propositional expansion of the skolemized problem. We also report about a first implementation of the procedure, which yields very interesting experimental results.

1 Introduction

Unquestionably, the most effective tools for solving a large class of industrial-scale problems (such as computer-aided design of integrated circuits [19], Planning [17], Model Checking for dynamic systems [5], Scheduling, Operations Research, and Cryptography, to name a few) are SAT solvers, which are search-based reasoning engines designed to decide the existence of models for propositional instances (PROP).

One step ahead of PROP, we encounter the more expressive language of quantified boolean formulas (QBFs), which adds the valuable possibility to quantify (universally or existentially) over the truth value of variables. Many of the problems mentioned above feature a far more handily QBF formulation, which is also (possibly) exponentially more succinct than the propositional one. For sure, by sticking to PROP we avoid worsening the decision complexity from NP to PSPACE. But, we also loose the expressive power of quantification, which not only provide a natural way to state relevant facts or rules, but could also be exploited during the solving process. At least in principle.

What really matters to applications is the capability of a reasoning engine to solve those problems that arise in practice. Hence, we ask: Is the balance between the above pros and cons favorable to QBF? Do quantified decision procedures add substantial value to the reasoning capabilities of purely propositional SAT solvers? The answer is: *not yet*. QBF is a promising formalism, but substantial improvements in decision procedures are expected before its potential can be unleashed to applications [1, 21, 4].

The observations above motivates this paper, in which we describe a new solving paradigm that captures the added value of quantified reasoning. A twofold novelty is introduced. On the one hand, we reinterpret the Skolem theorem to reassess quantified reasoning as a quantifier-free, propositional reasoning over a purposely designed

[§] This work was supported by PAT (*Provincia Autonoma di Trento*, Italy), grant n. 3248/2003.

symbolic representation. We show how this allows to mix (1) the inference power of quantified reasoning, (2) the strength of many well known SAT techniques, and (3) the classical search-based decision procedures for QBF. On the other hand, several powerful techniques for automated reasoning are arranged within a coherent framework in a way that is advantageous and instrumental in realizing the just mentioned approach.

The essential component of our construction traces back to the Twenties (the Skolem theorem [31]). Following the timeline, we capitalize on the seminal contributions to propositional theorem proving from the Sixties (DPLL algorithms [9, 10]). Then, a compact formalism from the Eighties to reason on boolean functions [6] is employed. Effective quantified reasoning comes from the Nineties [7, 18]. In the same years, techniques to translate real-world problems into SAT arised [17, 19, 5] which are adapted to our case. Finally, symbolic representations for propositional problems gained attention in the last few years [8, 25, 28], and are largely useful here. These techniques are exercised together thank to a symbolic representation for the propositional expansion of skolemized QBF instances, also resorting to SAT-based reasoning when it pays back.

In the rest of this paper we introduce some preliminaries (Section 2), present symbolic skolemization (Section 3), discuss symbolic reasoning strategies (Section 4), analyze experimental results (Section 5), and give our concluding remarks (Section 6).

2 Preliminaries

We consider quantified boolean formulas in *prenex conjunctive normal form*[1], such as

$$f = \forall a \exists b \forall c \exists d. \ (\neg a \vee c \vee d) \wedge (\neg b \vee \neg d) \wedge (a \vee b \vee \neg d) \wedge (\neg a \vee b) \qquad (1)$$

where "$\forall a \exists b \forall c \exists d$" is called *prefix* and is followed by a conjunctive normal form (CNF) *matrix*, i.e. a propositional formula made up by conjuncting clauses, each clause being a disjunction of literals (a variable or a negated variable). More in general, we consider formulas $\mathcal{Q}_1 V_1 \mathcal{Q}_2 V_2 \ldots \mathcal{Q}_n V_n. \mathcal{M}$ where $\mathcal{Q}_i \in \{\forall, \exists\}$, $\mathcal{Q}_i \neq \mathcal{Q}_{i+1}$, and the matrix $\mathcal{M}$ has variables $var(\mathcal{M}) = \cup_{i=1}^{n} V_i$ ($V_i \cap V_j = \emptyset$ for $i \neq j$). Variables $v \in V_i$ are said to be existentially (universally) quantified if $\mathcal{Q}_i = \exists$ ($\mathcal{Q}_i = \forall$). The set of existentially (universally) quantified variables in a QBF f is denoted by $var_\exists(f)$ ($var_\forall(f)$). The universal depth $\delta(v)$ of an existential variable $v \in V_i$ is the number of universal variables dominating v: $\delta(v) = \sum_{Q_j = \forall, j<i} |V_j|$. For clauses, it is $\delta(\Gamma) = max_{v \in var_\exists(\Gamma)} \delta(v)$.

We use lowercase (uppercase) roman letters for propositional variables (clauses), and greek letters for values in the boolean space $\mathfrak{B} = \{0, 1\}$ and vectors in $\mathfrak{B}^n$. For example: $\Psi = \langle \psi_1, \ldots, \psi_n \rangle \in \mathcal{I} \subseteq \mathfrak{B}^n$. The complement of $\mathcal{I} \subseteq \mathfrak{B}^n$ is denoted by $\overline{\mathcal{I}}$.

Double negation on literals is disallowed: $\neg\neg l$ is rewritten as l. We use exclusive-or to build literals out of variables: $\varphi \otimes v$ means v when $\varphi = 0$, and $\neg v$ when $\varphi = 1$. Given a literal $l = \varphi \otimes v$, we pose $var(l) = v$. An *assignment* is a consistent set of literals (i.e. a set S such that $\nexists l \in S | \neg l \in S$). We denote by $C * l$ the result of applying the assignment $\{l\}$ to the clause C. C is unchanged if $var(l) \notin var(C)$, is *subsumed* if $l \in C$, and *resolves* to $C \setminus \{\neg l\}$ if $\neg l \in C$. This notion is extended to sets of clauses and literals: $f * A$ is the formula resulting from applying A to each clause in f.

Finally, *forall reduction* is a model preserving transformation for QBFs that consists in removing from each clause C all the universal literals $\pi \otimes v \in C$ with $v \in V_i$ such that $var_\exists(C) \cap V_j = \emptyset$ for every $j > i$. We always consider forall-reduced formulas.

[1] This choice causes no loss of generality and is shared by all the available encodings of real-world problems [15]. However, it might be responsible for an increase in proof complexity.

3 Symbolic Propositional Skolemization for QBFs

We leverage the Skolem theorem to map any QBF instance onto a satisfiability equivalent SAT instance featuring a very compact symbolic representation. We interleave the development of the general method with the presentation of a running example.

3.1 Propositional Skolemization

The Skolem theorem [31] 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 (see [12] for a survey). Existential quantifiers are eliminated by replacing the variables they bind with *Skolem functions* whose definition domains are appositely chosen to preserve satisfiability. We empoly an *outer* form of skolemization [26], in which the function introduced for $e \in var_\exists(f)$ depends on all the universal variables that have e in their scope (for prenex formulas: all the universal variables to the left of e in the prefix).

Functions have no direct representation in $PROP$, but their *definability* can be captured at the expense of a possibly exponential blowup in the size of the instance.

We adopt a three-step *propositional skolemization* for a QBF instance f.

1. Translation of f into a satisfiability equivalent FOL instance $FOL(f)$.
2. Application of the Skolem theorem to $FOL(f)$ to obtain a (satisfiability preserving) FOL instance $Sk(FOL(f))$ with no existential quantifier.
3. Translation of $Sk(FOL(f))$ into an equivalent SAT instance $Prop(Sk(FOL(f)))$.

The first step is a slight rephrase of the problem, but it allows us to plainly capture the intuition behind propositional skolemization. Skolem funtions leverage the existence of two semantics levels in FOL, namely the level of *predicates* and the level of *terms*. Skolem functions are terms that are substituted for other terms (the existential variables) as arguments of predicates. QBF and $PROP$ lack the formal mechanisms necessary to cope with those two levels. They just feature the predicate level, though this is obfuscated by their variable-oriented syntax. To uncover such level, we introduce a FOL unary predicate $\mathbf{p}$ defined over the boolean space $\mathfrak{B}$, and interpreted as $\mathbf{p}(1) = TRUE$, $\mathbf{p}(0) = FALSE$, and restrict the domain of interpretation of every variable to be the boolean space as well. This immediately allows us to rewrite every QBF as a satisfiability equivalent FOL formula. For example, by rewriting the QBF (1) we obtain

$$\begin{aligned} f' = FOL(f) = \forall a \exists b \forall c \exists d.\ & (\neg\mathbf{p}(a) \vee \mathbf{p}(c) \vee \mathbf{p}(d)) \wedge (\neg\mathbf{p}(b) \vee \neg\mathbf{p}(d)) \\ & \wedge(\mathbf{p}(a) \vee \mathbf{p}(b) \vee \neg\mathbf{p}(d)) \wedge (\neg\mathbf{p}(a) \vee \mathbf{p}(b)) \end{aligned}$$

In the second step we eliminate existential quantifiers by substituting to each existential variable v a Skolem function s^v depending on the proper subset of dominating universal quantifiers. We obtain a satisfiability-equivalent purely universal formula.

$$\begin{aligned} Sk(f') = \forall a \forall c.\ & (\neg\mathbf{p}(a) \vee \mathbf{p}(c) \vee \mathbf{p}(s^d(a,c))) \wedge (\neg\mathbf{p}(s^b(a)) \vee \neg\mathbf{p}(s^d(a,c))) \\ & \wedge(\mathbf{p}(a) \vee \mathbf{p}(s^b(a)) \vee \neg\mathbf{p}(s^d(a,c))) \wedge (\neg\mathbf{p}(a) \vee \mathbf{p}(s^b(a))) \end{aligned} \qquad (2)$$

From a FOL point of view, existential quantifiers are simply disappeared. The dute we pay for this simplification is the loss of logical equivalence. From a higher-level point of view, we can predicate over the interpretation of terms and explicitly state what the Skolem theorem implicitly says when it reduces the satisfiability of $FOL(f)$ to the satisfiability of $Sk(FOL(f))$, i.e. that each *inner* existential FOL quantification over v has been substituted by an *outer* higher-order existential quantification over s^v (over the existence of a proper interpretation for the Skolem terms). Informally:

$$\forall a \exists b \forall c \exists d.\ f(a,b,c,d) \quad \overset{SAT}{\Longleftrightarrow} \quad [\exists s^b \exists s^d] \forall a \forall c.\ f(a, s^b(a), c, s^d(a,c))$$

In the third step (translation to $PROP$), the actual work is done. It amounts to *flatten* the two semantics levels introduced above onto one single propositional level. This transformation is made easy by the constructive property that for every formula $Sk(FOL(f))$ with $f \in QBF$ both the predicate-level and the term-level interpretations map boolean spaces onto boolean values. We join their definition spaces and interpretation functions, and give an inductive translation procedure from $Sk(FOL(f))$ to $PROP$.

The only non-trivial piece of work consists of building a CNF propositional representation for every (possibly negated) Skolem term. As a constructive consequence of steps 1-2, every Skolem function $s(a_1, a_2, \ldots, a_n)$ we manage is a relation over $\mathfrak{B}^{n+1}$ that maps $\mathfrak{B}^n$ onto $\mathfrak{B}$. Each one is completely specified by 2^n boolean parameters giving the truth value of the function on each point of its domain, so 2^{2^n} different Skolem n-ary functions exist. Let us denote by s_Ψ the boolean parameter that represents the truth value of a boolean n-ary function s evaluated in $\Psi = \langle \psi_i, \psi_2, \ldots, \psi_n \rangle \in \mathfrak{B}^n$. We directly obtain a CNF propositional version for $s(a_1, a_2, \ldots, a_n)$ as follows:

$$Prop(\varphi \otimes s(a_1, \ldots, a_n)) \doteq \bigwedge_{\Psi \in \mathfrak{B}^n} (\varphi \otimes s_\Psi) \vee \neg(\psi_1 \otimes a_1) \vee \neg(\psi_2 \otimes a_2) \vee \cdots \vee \neg(\psi_n \otimes a_n)$$

For example, the Skolem terms $\neg s^b(a)$ and $s^d(a,c)$ are translated as $Prop(\neg s^b(a)) = (\neg s^b_0 \vee a) \wedge (\neg s^b_1 \vee \neg a)$, and $Prop(s^d(a,c)) = (s^d_{11} \vee \neg a \vee \neg c) \wedge (s^d_{10} \vee \neg a \vee c) \wedge (s^d_{01} \vee a \vee \neg c) \wedge (s^d_{00} \vee a \vee c)$. So, $Prop(s^d(a,c))$ may be seen as a function mapping a point $\langle \alpha, \gamma \rangle \in \mathfrak{B}^2$ onto the proper value $s^d_{\alpha\gamma} = s^d(\alpha, \gamma)$, and the same for $Prop(\neg s^b(a))$.

The next step extends the translation from terms to predicates. Let us first consider a clause containing only one existentially quantified variable e with polarity φ:

$$\forall u_1 \forall u_2 \cdots \forall u_n \exists e.\ (\pi_1 \otimes u_{i_1}) \vee (\pi_2 \otimes u_{i_2}) \vee \cdots \vee (\pi_r \otimes u_{i_r}) \vee (\varphi \otimes e)$$

where $u_1, u_2, \ldots, u_n$ are all the universal variables dominating e, while a subset $u_{i_1}, u_{i_2}, \ldots, u_{i_r}$, $r \leq n$ of such variables appears in the clause with polarities $\pi_1, \pi_2, \ldots, \pi_r$. By substituting for e the expansion $Prop(\varphi \otimes s(u_1, \ldots, u_n))$ of the Skolem function $s : \mathfrak{B}^n \rightarrow \mathfrak{B}$ defined by the 2^n parameters $\{s_{0\ldots00}, s_{0\ldots01}, \cdots, s_{1\ldots11}\}$, we obtain:

$$\begin{aligned}
&\exists s_{0\ldots00} \exists s_{0\ldots01} \cdots \exists s_{1\ldots11} \\
&\qquad \forall u_1 \forall u_2 \cdots \forall u_n \\
&\qquad\qquad (\pi_1 \otimes u_{i_1}) \vee (\pi_2 \otimes u_{i_2}) \vee \cdots \vee (\pi_r \otimes u_{i_r}) \vee \\
&\qquad\qquad\qquad \vee \left(\textstyle\bigwedge_{\Psi \in \mathfrak{B}^n} (\varphi \otimes s_\Psi) \vee \neg(\psi_1 \otimes u_1) \vee \cdots \vee \neg(\psi_n \otimes u_n) \right)
\end{aligned}$$

As a consequence of the semantics flattening we have performed, the “meta” existential quantifier over an n-ary Skolem function has been transformed into a set of 2^n outer existential quantifiers. In the worst case, we have to distribute the disjunction over all the clauses in the last term, thus obtaining 2^n clauses. Fortunately, some (many) of those clauses are trivially satisfied by complementary literals. In particular, whenever $\psi_{i_j} = \pi_j$ for at least one $j \in \{1, \ldots, r\}$, the clause is satisfied, so that we get only $2^{n-r} = 2^{\delta(e)-r}$ clauses. Moreover, skolemized clauses no longer contain existential variables dominated by universal variables, hence all the universal literals are *forall reducible*. As a result of these two properties, we obtain the set of unit clauses:

$$\exists s_{0\dots00}\exists s_{0\dots01}\cdots\exists s_{1\dots11}.\ \bigwedge_{\Psi\in\mathcal{I}}\varphi\otimes s_{\Psi},\quad \text{where } \mathcal{I}=\{\Psi\in\mathfrak{B}^n|\forall j.\psi_{i_j}\neq\pi_j\}$$

In the general case we have clauses containing m existential variables $\{e_1, e_2, \dots, e_m\}$ with $\delta(e_1)\le\delta(e_2)\le\dots\le\delta(e_m)$ and polarities $\varphi_1,\dots,\varphi_m$, where each e_i is dominated by a set $\cup_{j=0}^{i}U_j$ of universal variables. Each clause also contains a possibly empty subset of universal literals $\{\pi_k\otimes u_k, k=i_{j-1}+1,\dots,i_j\}\subseteq U_j$ for each $j=1,\dots,m$, with $i_0=0$. The general shape for the clause is

$$\begin{aligned}\forall U_1\exists e_1\cdots\forall U_m\exists e_m.\quad & (\pi_1\otimes u_{i_1})\vee\cdots\vee(\pi_{j_1}\otimes u_{i_{j_1}})\vee\ (\varphi_1\otimes e_1)\vee\\ & (\pi_{j_1+1}\otimes u_{i_{j_1+1}})\vee\cdots\vee(\pi_{j_2}\otimes u_{i_{j_2}})\vee\ (\varphi_2\otimes e_2)\vee\\ & \qquad\vdots\\ & (\pi_{j_{m-1}+1}\otimes u_{i_{j_{m-1}+1}})\vee\cdots\vee(\pi_{j_m}\otimes u_{i_{j_m}})\vee\ (\varphi_m\otimes e_m)\end{aligned}\tag{3}$$

By (a) propositionally skolemizing all the existential variables in such clause, and (b) applying forall reduction to all the variables in $\cup_{j=0}^{m}U_j$, we obtain:

$$\exists S_1\cdots\exists S_m.\ \bigwedge_{\substack{\Psi\,\in\,\mathfrak{B}^{\delta_m}\\ \forall j.\psi_{i_j}\neq\pi_j}}(\varphi_1\otimes s^{(1)}_{\Psi|_{\delta_1}})\ \vee\ (\varphi_2\otimes s^{(2)}_{\Psi|_{\delta_2}})\ \vee\ \cdots\ \vee\ (\varphi_m\otimes s^{(m)}_{\Psi|_{\delta_m}})\tag{4}$$

where $\Psi|_k$ denotes the k-bit long prefix of Ψ, $\delta_i=\delta(e_i)$, the boolean parameter $s^{(i)}_{\Psi'}$ represents the truth value over $\Psi'=\Psi|_{\delta_i}\in\mathfrak{B}^{\delta_i}$ of the Skolem function $s^{(i)}$ introduced for e_i, and $S_i=\{s^{(i)}_{\Psi}|\Psi\in\mathfrak{B}^{\delta_i}\}$. The abstraction operator "|" generalizes to sets as $\mathcal{I}|_k\doteq\{\langle\psi_1\ \dots,\psi_k\rangle|\exists\langle\psi_1,\dots,\psi_k,\dots,\psi_n\rangle\in\mathcal{I}\}$ with $\mathcal{I}\subseteq\mathfrak{B}^n$ and $k\le n$.

We denote by $PropSk(\cdot)$ the function that applied to a generic QBF clause represented by Expression (3) yields the result of our three-step translation, i.e. the set of clauses represented by Expression (4). The cardinality of this clause set is $2^{\delta(e_m)-j_m}$.

To translate an entire formula, we observe that Skolem terms are introduced once per variable. So, the propositional skolemization of any formula is obtained by joining together the skolem clauses obtained out of each QBF clause, always re-using the same parameters on the same existential variable. The overall procedure defines a satisfiability-preserving mapping $PropSk : QBF \longrightarrow PROP$ between the original QBF space and a purely propositional space. For a QBF f with $var_{\exists}(f)=\{e_1,\dots,e_m\}$, the $PROP$ instance is defined over the set of fresh variables $\{s^{(i)}_{\Psi}, i=1,\dots,m,\Psi\in\mathfrak{B}^{\delta(e_i)}\}$.

As an example, by propositionally skolemizing (1) we obtain

$$\begin{aligned}\exists s^b_0\exists s^b_1\exists s^d_{00}\exists s^d_{01}\exists s^d_{10}\exists s^d_{11}.\ & (s^d_{01})\wedge(\neg s^b_0\vee\neg s^d_{00})\wedge(\neg s^b_0\vee\neg s^d_{01})\wedge(\neg s^b_1\vee\neg s^d_{10})\\ & \wedge(\neg s^b_1\vee\neg s^d_{11})\wedge(s^b_1\vee\neg s^d_{10})\wedge(s^b_1\vee\neg s^d_{11})\wedge(s^b_0)\end{aligned}\tag{5}$$

If (and only if) we find a model for (5) we are entitled to conclude that (2) is satisfiable, so that (1) evaluates to $TRUE$. Not only we are ensured that a proper interpretation for the Skolem functions s^b and s^d do exist to satisfy the formula, but we have explicitily *computed* such an interpretation. Every model for (5) gives us the desired truth value of acceptable skolem functions over each point of their domains.

3.2 Symbolic Representation

The term "symbolic representation" has a broad AI-related sense, but it has been used with a much more specific meaning in the realm of model checking (MC). According to MC's usage of the word, a symbolic representation is one that allows to shift from *explicit* MC techniques—where each state of a system to be checked is individually represented and manipulated—to *symbolic* MC approaches—where data structures are employed that allow to compactly and implicitly represent (possibly huge) sets of states, and also to reason about them as a whole. We adopt MC's viewpoint here, as we are interested in symbolically representing and manipulating sets of clauses.

This interest originates in the observation that $PropSk(f)$ may be exponentially larger than f. Without some powerful tool for compactely representing and managing propositional skolemizations, not only it may be unfeasible to solve the resulting SAT instances, but they might not even fit into the memory of any real machine.

Related approaches exist in the literature (see Section 6), but we have to manage a very special case here, and we want to profit from its structure. In particular, we are only interested in representing clause-sets coming from propositional skolemization of QBF formulas, with a representation that is closed under the operations we define in the next section. Our representation employes one single *symbolic clause* to compactly represent the whole clause set described by the Expression (4) w.r.t. the QBF clause C described in Expression (3). We need to memorize three pieces of information:

1. The list $\Gamma = [\varphi_1 \otimes e_1, \ldots, \varphi_m \otimes e_m]$ of existential literals in the originating clause.
2. The set of indexes $\mathcal{I} = \{\Psi \in \mathfrak{B}^{\delta(e_m)} \mid \forall j. \psi_{i_j} \neq \pi_j\}$.
3. The list $[\delta(e_1), \ldots, \delta(e_m)]$ of the universal depths of each existential literal.

The information in Item 3 is not related to a single clause. Rather, it is an attribute of the formula as a whole that only depends on the prefix, and that needs to be represented once per formula. By contrast, the couple $\langle \Gamma, \mathcal{I} \rangle$ actually defines a symbolic clause $Symb(C)$ which we compactly denote by writing $\Gamma_{\mathcal{I}}$. The $Symb(\cdot)$ transformation is readily extended to QBF instances as $Symb : QBF \longrightarrow PROP_{SYMB}$, where $PROP_{SYMB}$ denotes the space of symbolic propositional instances. It is

$$Symb(\forall U_1 \exists e_1 \cdots \forall U_m \exists e_m . M) \doteq \exists [e_1]_{\delta_1} \cdots \exists [e_m]_{\delta_m} . \bigwedge_{C \in M} Symb(C) \tag{6}$$

where $\exists [e_1]_{\delta_1} \cdots \exists [e_m]_{\delta_m}$ is a *symbolic prefix* mentioning a *symbolic variable* $[e_i]_{\delta_i}$ for each original existential variable e_i at universal depth $\delta_i = \delta(e_i)$.

For example, the symbolic representation of the QBF formula (1) is:

$$\mathcal{F} = \exists [b]_1 \exists [d]_2 . \, [d]_{\{10\}} \wedge [\neg b, \neg d]_{\{00,01,10,11\}} \wedge [b, \neg d]_{\{00,01\}} \wedge [b]_{\{1\}} \tag{7}$$

Each symbolic clause is made up of *symbolic literals*, that we represent as symbolic unit clauses. For example, the clause $[b, \neg d]_{\{00,01\}}$, under the prefix $\exists [b]_1 \exists [d]_2$, is made up by the symbolic literals $[b]_{\{0\}}$ and $[\neg d]_{\{00,01\}}$. A symbolic literal $[\varphi \otimes e]_{\mathcal{I}}$ belongs to a symbolic clause $\Gamma_{\mathcal{J}}$, written $[\varphi \otimes e]_{\mathcal{I}} \in \Gamma_{\mathcal{J}}$, when $\varphi \otimes e \in \Gamma$ and $\mathcal{I} \subseteq \mathcal{J}|_{\delta(e)}$. As opposite to symbolic objects, the standard propositional elements are called *ground* objects. For example, the ground literals $\neg d_{00}$ and $\neg d_{01}$ belong to the symbolic literal $[\neg d]_{\{00,01\}}$, while the ground clauses $b_0 \vee \neg d_{01}$ and $b_0 \vee \neg d_{00}$ belongs to $[b, \neg d]_{\{00,01\}}$.

Symbolic formulas have both a *symbolic size* and a *ground size*. The symbolic size of $\mathcal{F}$ is the number of symbolic clauses (or literals) in the formula. The ground size is the number of clauses (or literals) in $Prop(\mathcal{F})$. So, the symbolic size (number of clauses) for a symbolic formula $\mathcal{F}$ is $|\mathcal{F}|_{symb} = \sum_{\Gamma_\mathcal{I} \in \mathcal{F}} |\Gamma|$, while its ground size is $|\mathcal{F}|_{ground} = \sum_{\Gamma_\mathcal{I} \in \mathcal{F}} |\mathcal{I}|$. For example, the formula (7) has symbolic size equal to 4 and ground size equal to 8. The ground size is always greater than the symbolic size, as each symbolic clause represents at least one ground clause.

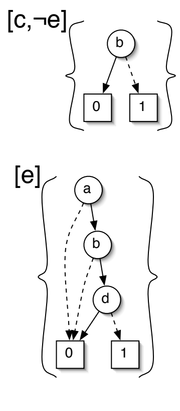
Symbolic formulas exhibit three appealing properties: (1) they preserve the satisfiability of the originating QBF instance, (2) they are compactly representable, and (3) they can be efficiently manipulated to perform deductions. We here consider the first two properties, and delay the discussion on the third one until the next section.

Semantics for symbolic formulas. We define an evaluation mechanism for symbolic formulas based on the standard evaluation of their propositional expansion. According to Expression (4), we can re-gain the ground meaning of $\Gamma_\mathcal{I} = [\varphi_1 \otimes e_1, \ldots, \varphi_m \otimes e_m]_\mathcal{I}$ under the relevant prefix $\mathcal{P} = \exists[e_1]_{\delta_1} \cdots \exists[e_m]_{\delta_m}$ through a function $Prop$ defined as

$$Prop(\mathcal{P}, \Gamma_\mathcal{I}) \doteq \bigwedge_{\Psi \in \mathcal{I}} \varphi_1 \otimes s^{(1)}_{\Psi|_{\delta_1}} \vee \ldots \vee \varphi_m \otimes s^{(m)}_{\Psi|_{\delta_m}} \tag{8}$$

This function is extended to a symbolic formula $\mathcal{F}$ with matrix $\mathcal{M}$ and prefix $\mathcal{P}$ by posing $Prop(\mathcal{F}) \doteq \bigwedge_{\Gamma_\mathcal{I} \in \mathcal{M}} Prop(\mathcal{P}, \Gamma_\mathcal{I})$. In particular, a consistent set of symbolic literals $\{[e_1]_{\mathcal{I}_1}, \ldots, [e_k]_{\mathcal{I}_k}\}$ is a model for $\mathcal{F} = \mathcal{P}.\ \mathcal{M} = Symb(f), f \in QBF$ iff the ground assignment $\cup_{j=1,\ldots,k} Prop(\mathcal{P}, [e_j]_{\mathcal{I}_j})$ is a model for $Prop(Symb(f))$. By construction, it is $Prop(Symb(f)) = PropSk(f)$, hence the QBF f evaluates to $TRUE$ iff $Symb(f)$ is satisfiable. For example, the $Prop$ function applied to (7) yields (5).

Compact representation. The (possible) exponential blowup in every symbolic clause $\Gamma_\mathcal{I}$ has been purposely confined to the cardinality of $\mathcal{I}$. We pursue compactness for its symbolic representation, notwithstanding its ground size, by employing a second layer of abstraction, consisting in the compact representation of $\mathcal{I}$ by means of *reduced ordered binary decision diagrams* (BDDs) defined over the set of variables $var_\forall(f)$.



According to the semantics of BDDs, an entire set $\mathcal{I} = \{\Psi \in \mathfrak{B}^{\delta(e_m)} \mid \forall j. \psi_{i_j} \neq \pi_j\}$ is represented by a single linear-sized BDD (in m) requiring one internal node for each universal variable in the originating clause. The whole symbolic representation has a linear size w.r.t. the number of literals in the originating QBF clause. The picture aside depicts our representation of the skolemized version of $\forall a \forall b \exists c \forall d \exists e.\ (b \vee c \vee \neg e) \wedge (\neg a \vee \neg b \vee d \vee e)$. As an additional source of compactness, we notice that BDDs are semantically canonical representations, so they share *at least* all the representations for QBF clauses with the same universal literals. As we produce only one symbolic clause out of each QBF clause, the representation of $Symb(f)$ enlarges at most linearly with $|f|$. However, this *only holds for the initial representation*. The symbolic size may increase as a consequence of the symbolic inferences described in the next section.

4 Reasoning on Propositional Skolemizations

The evaluation of the original QBF instance has been restated as a satisfiability test on the symbolically represented existential instance $Symb(f)$. The peculiar structure of $Symb(f)$ allows to attack the SAT problem from three different perspectives, each one featuring specific strengths. We describe such methods in the subsequent three sections. Far from being mutually exclusive, those three strategies can be used in a synergic way, so that each one contributes at its best towards the common goal (see Section 4.3).

4.1 Ground reasoning

The original QBF f can be evaluated by explicitly constructing $Prop(Symb(f))$ and solving it via state-of-the-art SAT solvers (they are very efficient on QBF-derived instances). We resort to this option only when the ground instance is affordable[2], which is not the case for many real-world problems. Yet, the reduced problems generated as described in Section 4.2, 4.3 are eventually small enough to be solved this way.

Altought theorically straightforward, the computation of $Prop(Symb(f))$ deserves a lot of attention on the practical side, due to the (possibly) large number of (possibly) huge SAT instances generated out of each QBF formula. Groundization is made up of two steps: (1) generation of the *ground space* and (2) generation of the ground clauses. The latter step is executed according to Expression (8). The former constructs a mapping between the *structured* namespace of symbolic literals and a *flat*, SAT-solver friendly namespace for ground literals. It amounts to associate a unique positive integer to each ground variable that belongs to at least one clause in the current symbolic formula (and to them only). To prevent the SAT solver from suffering unnecessarily large data structures, the set of variable codes generated for the formula as a whole should be composed of all and only the integers in the interval $[1, n]$, for some sufficiently large n. In essence, we need a partial, efficiently invertible function $V_{map} : D_{\exists} \times D_{\forall} \rightarrow [1, n]$ where $D_{\exists} = var_{\exists}(f)$, $D_{\forall} = \mathfrak{B}^{|var_{\forall}(f)|}$, and n just suffices to allow bijection.

4.2 Symbolic reasoning

We define some *symbolic inference rules* over $PROP_{SYMB}$ to directly manipulate $Symb(f)$ while preserving the satisfiability of $Prop(Symb(f))$. As opposed to ground reasoning, the emphasis is on designing symbolic versions of the standard inference rules that work without expanding symbolic objects to ground objects. In essence, it is a matter of defining how the basic steps (subsumption, resolution, assignment, substitution) can be performed at a purely symbolic level on sets of ground clauses at once.

Complete refutation strategies—such as those based on SL, linear, or directional resolution—could be employed in principle. However, efficient and easy-to-implement forms of incomplete reasoning exist that capture many inferences relevant to QBF-derived instances. Even if the rules adopted are not refutationally complete, the computation of their deductive closure *normalizes* the instance. So, a satisfiability-equivalent, symbolic output formula with a (much) smaller ground size than the original one is generated, and other complete methods can safely work on such simplified version.

[2] By *affordable* we mean that the instance can be decided without running out of memory. Affordability thus depends on the SAT engine employed and on the available amount of memory.

The central step towards symbolic reasoning amounts to extend the star operator. Formerly absent empty clause-sets $\mathit{\Gamma}_\emptyset$ may result, which are eliminated from the formula.

$$\mathit{\Gamma}_{\mathcal{I}} * [l]_{\mathcal{J}} = \begin{cases} \mathit{\Gamma}_{\mathcal{I}\cap\overline{\mathcal{J}}} & \text{when } l \in \mathit{\Gamma} \\ \mathit{\Gamma}_{\mathcal{I}\cap\overline{\mathcal{J}}} \wedge \mathit{\Gamma}'_{(\mathcal{I}\cap\mathcal{J})|_{\delta(\mathit{\Gamma}')}} & \text{with } \mathit{\Gamma}' = \mathit{\Gamma} \setminus \{\neg l\}, \text{when } \neg l \in \mathit{\Gamma} \\ \mathit{\Gamma}_{\mathcal{I}} & \text{otherwise} \end{cases} \tag{9}$$

The efficiency of symbolic reasoning thus stems from the structured nature of the representation, which takes universal reasoning apart form existential reasoning. BDD operations conveniently deal with the former, list-based representations with the latter.

We now exemplify four (incomplete) symbolic rules that are highly effective on average and can be implemented rather efficiently. In particular, notice that it is easy to symbolically extract both pure literals and unit clauses. Let us consider the formula

$$\begin{array}{r} \exists a \forall b \exists c \forall de \exists fgh \forall i \exists l.\ (\neg c \vee a) \wedge (\neg a \vee \neg g) \wedge (\neg e \vee h) \wedge (c \vee \neg e \vee g \vee \neg h) \\ \wedge(\neg b \vee d \vee \neg f \vee l) \wedge (\neg e \vee f \vee g) \wedge (i \vee \neg c \vee \neg h \vee d \vee \neg l) \end{array} \tag{10}$$

and its symbolic matrix $\mathcal{M}$ (under the prefix $\exists[a]_0\exists[c]_1\exists[f]_3\exists[g]_3\exists[h]_3\exists[l]_4$):

$$\begin{array}{c} [\neg c, a]_{\{0,1\}} \wedge [\neg a, \neg g]_{\{0,1\}^3} \wedge [h]_{\{001,011,101,111\}} \wedge [c, g, \neg h]_{\{001,011,101,111\}} \wedge \\ [\neg f, l]_{\{1000,1001,1010,1011\}} \wedge [g, f]_{\{001,011,101,111\}} \wedge [\neg c, \neg h, \neg l]_{\{0000,1000,0010,1010\}} \end{array}$$

The simplest rule is the *symbolic unit clause propagation* (SUCP). It builds on top of the observation that each symbolic unit clause $[\gamma]_{\mathcal{I}}$ in the formula represents a set $\{\gamma_i | i \in \mathcal{I}\}$ of ground literals. All of them need to be assigned to avoid contradictions. These assignments are performed all-at-once by the star operator. The only unit clause in our symbolic formula is $[h]_{\{001,011,101,111\}}$. By assigning this literal we obtain

$$\begin{array}{c} [\neg c, a]_{\{0,1\}} \wedge [\neg a, \neg g]_{\{0,1\}^3} \wedge [c, g]_{\{001,011,101,111\}} \wedge [\neg f, l]_{\{1000,1001,1010,1011\}} \\ \wedge [g, f]_{\{001,011,101,111\}} \wedge [\neg c, \neg l]_{\{0000,1000\}} \wedge [\neg c, \neg h, \neg l]_{\{0010,1010\}} \end{array}$$

The next rule we apply is the *symbolic pure literal elimination* (SPLE). It does what we would expect from the standard rule, but performs its job in a purely symbolic manner, by (a) constructing a complete symbolic representation of the set of every pure ground literal, and (b) applying the resulting symbolic literals to the formula. The pure literals on v are $[v]_{\mathcal{I}^+ \setminus (\mathcal{I}^+ \cap \mathcal{I}^-)}$ and $[\neg v]_{\mathcal{I}^- \setminus (\mathcal{I}^+ \cap \mathcal{I}^-)}$, where $\mathcal{I}^+ = \cup_{[v]_{\mathcal{I}} \in \mathcal{M}} \mathcal{I}$ and $\mathcal{I}^- = \cup_{[\neg v]_{\mathcal{I}} \in \mathcal{M}} \mathcal{I}$. The pure literals in our example are $[f]_{\{001,011,111\}}$, $[\neg f]_{\{100\}}$, $[\neg g]_{\{000,010,100,110\}}$, $[\neg h]_{\{000,100\}}$, $[l]_{\{1001,1011\}}$, and $[\neg l]_{\{0000,0010\}}$, so we obtain

$$\begin{array}{c} [\neg c, a]_{\{0,1\}} \wedge [\neg a, \neg g]_{\{001,011,101,111\}} \wedge [c, g]_{\{001,011,101,111\}} \\ \wedge [\neg f, l]_{\{1010\}} \wedge [g, f]_{\{101\}} \wedge [\neg c, \neg l]_{\{1010\}} \end{array}$$

The next two rules only consider the subset of *binary* symbolic clauses, employing a graph-based approach similar to the one used to simplify standard propositional instances with many binary clauses [2]. We build a *symbolic implication graph* (SIG), which has two nodes labeled by $[a]_{\delta(a)}$ and $[\neg a]_{\delta(a)}$ for each existential variable a in the original QBF, and a couple of arcs $[\neg a]_{\delta(a)} \xrightarrow{\mathcal{I}} [b]_{\delta(b)}$ and $[\neg b]_{\delta(b)} \xrightarrow{\mathcal{I}} [a]_{\delta(a)}$ for each binary symbolic clause $[a, b]_{\mathcal{I}}$. So, unlike standard implication graphs, SIGs feature labeled arcs. The arcs originating from $[a, b]_{\mathcal{I}}$ are labeled by $\mathcal{I}$. Each symbolic arc $a \xrightarrow{\mathcal{I}} b$ represents a set of ground arcs $\{a_{\mathit{\Psi}|_{\delta(a)}} \longrightarrow b_{\mathit{\Psi}|_{\delta(b)}}, \mathit{\Psi} \in \mathcal{I}\}$ in the corresponding ground graph. The two rules we apply are as follows.

1. *Symbolic Hyper Binary Resolution* (SHBR). It enumerates all the resolution chains of symbolic binary clauses (via a depth-first, non-redundant traversal of the SIG), looking for *failed* literals, i.e. for literals $[a]_{\mathcal{I}}$ such that each $\neg a_{\Psi} \in [\neg a]_{\mathcal{I}}$ can be derived (via a finite number of resolution steps only involving binary clauses) as a consequence of the hypothesis a_{Ψ}. Each ground literal in $[a]_{\mathcal{I}}$ generates a contradiction, so we force the opposite symbolic assignment, shifting our attention onto $\mathcal{F} * [\neg a]_{\mathcal{I}}$. A literal $[a]_{\mathcal{I}}$ is failed if we encounter the following (portion of a) resolution path: $[a] \overset{\mathcal{I}_1}{\rightarrow} [a_1] \overset{\mathcal{I}_2}{\rightarrow} [a_2] \cdots \overset{\mathcal{I}_n}{\rightarrow} \neg[a]$, with $\mathcal{I} = (\cap_{j=1,\ldots,n}\mathcal{I}_j)|_{\delta(a)} \neq \emptyset$.
2. *Symbolic Equivalence Reasoning* (SER). It aims at identifying symbolic equivalences $[a] \overset{\mathcal{I}}{\leftrightarrow} [b]$, meaning that for all $\Psi \in \mathcal{I}$, $a_{\Psi|_{\delta(a)}} \leftrightarrow b_{\Psi|_{\delta(b)}}$ is a consequence of $Prop(\mathcal{F})$. It is easy to rewrite the substitution rule to apply all such equivalences at once, producing at most two symbolic clauses out of each clause involved in. To reduce the ground size of the formula we substitute $[a]$ for $[b]$ if $\delta(a) \leq \delta(b)$, and vice-versa. SER is performed by extracting all the strongly connected components (SCCs) from the SIG, temporarily discarding arc labels. Then, for each SCC we enumerate all its non-intersecting loops $[a] \overset{\mathcal{I}_1}{\rightarrow} [a_1] \overset{\mathcal{I}_2}{\rightarrow} [a_2] \cdots \overset{\mathcal{I}_n}{\rightarrow} [a]$ (let us suppose without loose of generality that $\delta(a) \leq \delta(a_i), i = 1, \ldots, n-1$), and apply the substitutions $[a] \overset{\mathcal{I}}{\leftrightarrow} [a_i]$, $i = 1, \ldots, n-1$, with $\mathcal{I} = (\cap_{j=1,\ldots,n}\mathcal{I}_j)|_{\delta(a)}$.

In our example, all the remaining symbolic clauses are binary. Notice that for the class of QBF instances with at most two existential literals per clause, the symbolic binary rules inherit completeness from their standard counterpart.

In the figure aside a fragment of the sample SIG is depicted. By SER we obtain $[c] \overset{\mathcal{I}}{\leftrightarrow} [\neg g]$ with $\mathcal{I} = \{\{001, 011, 101, 111\}\}\delta(c) < \delta(g)$, hence: $[a, \neg c]_{\{0,1\}} \wedge [\neg a, c]_{\{0,1\}} \wedge [\neg f, l]_{\{1010\}} \wedge [\neg c, f]_{\{101\}} \wedge [\neg c, \neg l]_{\{1010\}}$. Then, the failed literal $[c]_{\{1\}}$ can be deduced from $[c] \xrightarrow{\{101\}} [f] \xrightarrow{\{1010\}} [l] \xrightarrow{\{1010\}} [\neg c]$, so $[a, \neg c]_{\{0\}} \wedge [\neg a, c]_{\{0\}} \wedge [\neg a]_{\{1\}}$ $\wedge [\neg f, l]_{\{1010\}}$ remains. By assigning the pure literal $[l]_{1010}$ and the unit clause $[\neg a]_{\{1\}}$ we have $[a, \neg c]_{\{0\}} \wedge [\neg a, c]_{\{0\}}$, hence the empty formula by SER on $[c] \overset{\{0\}}{\leftrightarrow} [a]$.

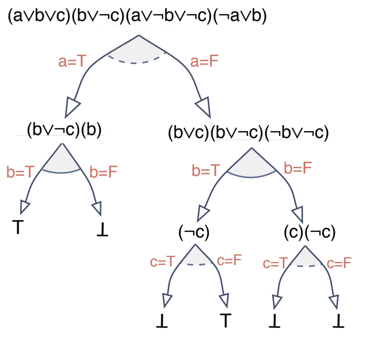
$$\begin{tikzcd}
 & {[c]_1} \arrow[dr, "{\{0,1\}}"] & \\
{[\neg g]_3} \arrow[ur, "{\{001,011,101,111\}}"] & & {[a]_0} \arrow[ll, "{\{001,011,101,111\}}"]
\end{tikzcd}$$

Applied until fixpoint, the above set of rules $\mathcal{R} = \{$SUCP,SPLE,SHBR,SER$\}$ defines a satisfiability preserving trasformation $Norm_{\mathcal{R}} : PROP_{SYMB} \rightarrow PROP_{SYMB}$.

4.3 Branching reasoning

In addition to symbolic and SAT reasoning, our representation fits well into search-based branching decision procedures. As far as QBFs are concerned, branching procedures extend the DPLL-approach [9] to the quantified case [7]. They look for models following the left-to-right order of the variables in the prefix during a depth-first visit of the semantic evaluation tree of the formula. Existential variables generate *or* nodes that disjunctively split the branch, universal quantifiers are associated to *and* nodes that split branches conjunctively. Each node n is labeled by the cofactored matrix $M * \Delta$ where Δ is the assignment on the path to n, while the root is labeled by the original matrix M.

A model, if one exists, is a subtree with all the leaves labeled by $\top$, extracted by choosing only one child for each existential node, and both children for conjunctive nodes. For example, the formula $\exists a \forall b \exists c.(a \vee b \vee c) \wedge (b \vee \neg c) \wedge (a \vee \neg b \vee \neg c) \wedge (\neg a \vee b)$ is decided to be false by visiting the tree reported aside and failing to extract any model.
Inspired by the above strategy, we build an evaluation procedure that mixes ground, symbolic, and branching reasoning. We just need to define the following projection operator.



$$\Gamma_{\mathcal{I}\downarrow_\alpha} \doteq \Gamma_{\mathcal{I}'} \text{ with } \alpha \in \mathfrak{B} \text{ and } \mathcal{I}' = \{\langle \psi_2, \ldots, \psi_{\delta(\Gamma)} \rangle | \langle \alpha, \psi_2, \ldots, \psi_{\delta(\Gamma)} \rangle \in \mathcal{I}\}$$

Projection is used for universal branching and is readily extended to formulas:

$$(\exists [e_1]_{\delta_1} \cdots \exists [e_m]_{\delta_m} . \mathcal{M})\downarrow_\alpha = \exists [e_1]_{\delta_1 - 1} \cdots \exists [e_m]_{\delta_m - 1} . (\mathcal{M}\downarrow_\alpha) \tag{11}$$

where $\mathcal{M}\downarrow_\alpha = \wedge_{\Gamma_\mathcal{I} \in \mathcal{M}} \Gamma_{\mathcal{I}\downarrow_\alpha}$. Existential branching is done according to Expression (9). The resulting decision procedure is reported below.

As far as splitting over existential variables is concerned, the purely existential nature of $Prop(\mathcal{F})$ makes the whole procedure more similar to search-based SAT solvers than to QBF decision procedures. By contrast, when the split is performed over universal variables, something conceptually different happens: the instance is partitioned into two completely disjoint existential sub-instances, according to (11).

The two base-cases do not deal with trivial sub-formulas. Well in advance, either symbolic reasoning (whenever the current sub-instance falls within its deductive power) or ground reasoning act as powerful look-ahead tools. The usual enhancements to branching procedures (backjumping, learning, heuristics, etc.) also apply.

```
Function symbEval(symbolic formula $\mathcal{F}$)
  begin
    $\mathcal{F}' \leftarrow Norm_\mathcal{R}(\mathcal{F})$;
    if $\mathcal{F}' = \emptyset$ then
      return TRUE;
    else if $\bot \in \mathcal{F}'$ then
      return FALSE;
    else
      if ($|\mathcal{F}'|_{ground}$ is affordable) then
        return SAT($prop(\mathcal{F}')$);
      else
        Let $\mathcal{F}'$ be $\exists [e_1]_{\delta_1} \cdots \exists [e_m]_{\delta_m} . \mathcal{M}$ ;
        if ($\delta_1 > 0$) then
          return symbEval ($\exists [e_1]_{\delta_1 - 1} \cdots \exists [e_m]_{\delta_m - 1} . \mathcal{M}\downarrow_0$) $and$
                 symbEval ($\exists [e_1]_{\delta_1 - 1} \cdots \exists [e_m]_{\delta_m - 1} . \mathcal{M}\downarrow_1$);
        else
          return symbEval ($\exists [e_2]_{\delta_2} \cdots \exists [e_m]_{\delta_m} . \mathcal{M} * [e_1]$) $or$
                 symbEval ($\exists [e_2]_{\delta_2} \cdots \exists [e_m]_{\delta_m} . \mathcal{M} * [\neg e_1]$);
  end
```

5 Implementation and Experimentation

We present a first implementation of our decision procedure and a preliminary experimental evaluation. The interested reader may find further details and a wider experimentation in [3]. The resulting solver—called sKizzo—is a 60k-line piece of object-oriented C code managing ROBDDs through the CUDD package [32], version 2.4.0, and performing SAT solving using zChaff [24], version 2004.5.13.

We focus on a subset of the non-random families of instances collected in the QBFLIB's archive [15]. Among the others, we consider (1) Rintanen's benhmarks [29], the first and best-known collection of QBF problems, made up of 47 instances divided into 5 families, obtained by encoding planning problems, (2) Ayari's benchmarks [1], made up of 72 instances divided into 5 families, obtained from real-world verification problems on circuits and protocol descriptions (these instances are quite challenging

	Symbolic size (clauses)			Ground size (clauses)			
Instance	*Before*	*After*	*Diff.*	*Before*	*After*	*Diff.*	Symb. time
Adder2-2-c	234	193	-18%	$1.0 \cdot 10^6$	$5.4 \cdot 10^5$	-46.0%	100%
Adder2-6-s	3,315	2,236	-33%	$1.8 \cdot 10^{12}$	$1.0 \cdot 10^6$	-99.9%	23%
Adder2-8-s	6,060	4,070	-33%	$1.0 \cdot 10^{16}$	$2.2 \cdot 10^7$	-99.9%	14%
BLOCKS3i.5.4	2,640	2,814	+7%	$4.0 \cdot 10^4$	$3.0 \cdot 10^4$	-25.0%	100%
BLOCKS3ii.5.2	1,886	2,095	+11%	$2.9 \cdot 10^4$	$2.1 \cdot 10^4$	-28.0%	100%
BLOCKS3iii.5	1,226	1,614	+32%	$1.9 \cdot 10^4$	$1.3 \cdot 10^4$	-32.0%	100%
CHAIN12v.13	486	0	-100%	$1.8 \cdot 10^6$	0	-100.0%	100%
CHAIN17v.18	861	0	-100%	$1.1 \cdot 10^8$	0	-100.0%	100%
CHAIN23v.24	1,443	0	-100%	$1.2 \cdot 10^{10}$	0	-100.0%	100%
cnt08	1,237	0	-100%	$6.1 \cdot 10^4$	0	-100.0%	100%
cnt08re	1,309	1,240	-5%	$6.5 \cdot 10^4$	$1.1 \cdot 10^4$	-83.0%	<1%
cnt12	2,505	0	-100%	$1.3 \cdot 10^6$	0	-100.0%	100%
cnt12re	2,733	2,820	+3%	$1.5 \cdot 10^6$	$2.6 \cdot 10^5$	-83.0%	<1%
flipflop-9-c	74,066	71,691	-3%	$9.4 \cdot 10^{12}$	$9.2 \cdot 10^{12}$	-2.0%	100%
flipflop-10-c	128,245	124,844	-3%	$1.3 \cdot 10^{14}$	$1.3 \cdot 10^{14}$	-1.0%	100%
flipflop-11-c	210,674	205,995	-2%	$1.7 \cdot 10^{15}$	$1.7 \cdot 10^{15}$	-1.0%	100%
impl04	32	0	-100%	$1.4 \cdot 10^2$	0	-100%	100%
impl12	96	0	-100%	$3.7 \cdot 10^4$	0	-100%	100%
impl20	160	0	-100%	$9.4 \cdot 10^6$	0	-100%	100%
k-branch-n-9	12,923	20,608	+59%	$2.1 \cdot 10^{18}$	$1.6 \cdot 10^{18}$	-23.8%	3%
k-branch-p-13	28,676	78,006	+172%	$3.7 \cdot 10^{24}$	$2.9 \cdot 10^{24}$	-21.6%	100%
k-d4-n-16	5,133	5,535	+8%	$4.2 \cdot 10^{22}$	$2.4 \cdot 10^{22}$	-42.9%	2%
k-d4-p-16	2,959	5,044	+70%	$4.7 \cdot 10^{17}$	$3.1 \cdot 10^{17}$	-34.0%	100%
mutex-4-s	362	0	-100%	$1.9 \cdot 10^7$	0	-100%	100%
mutex-8-s	834	367	-56%	$2.9 \cdot 10^{12}$	$3.5 \cdot 10^4$	-99.9%	70%
TOILET10.1.iv.20	3,466	3,326	-4%	$2.1 \cdot 10^4$	$7.4 \cdot 10^3$	-64.8%	55%
TOILET16.1.iv.32	10,495	8,175	-22%	$5.6 \cdot 10^4$	$8.6 \cdot 10^3$	-84.6%	72%
toilet-a-08-01.11	3,109	1,069	-66%	$6.0 \cdot 10^4$	$2.7 \cdot 10^4$	-55.0%	3%
toilet-c-10-01.14	1,974	1,874	-5%	$7.5 \cdot 10^3$	$4.0 \cdot 10^3$	-46.6%	1%
toilet-g-20-01.2	460	0	-100%	$1.1 \cdot 10^3$	0	-100.0%	100%
tree-exa2-40	51	1	-100%	$5.6 \cdot 10^{14}$	1	-100%	100%
tree-exa10-30	58	0	-100%	58	0	-100%	100%

Table 1. The effect of symbolic reasoning over the size of instances.

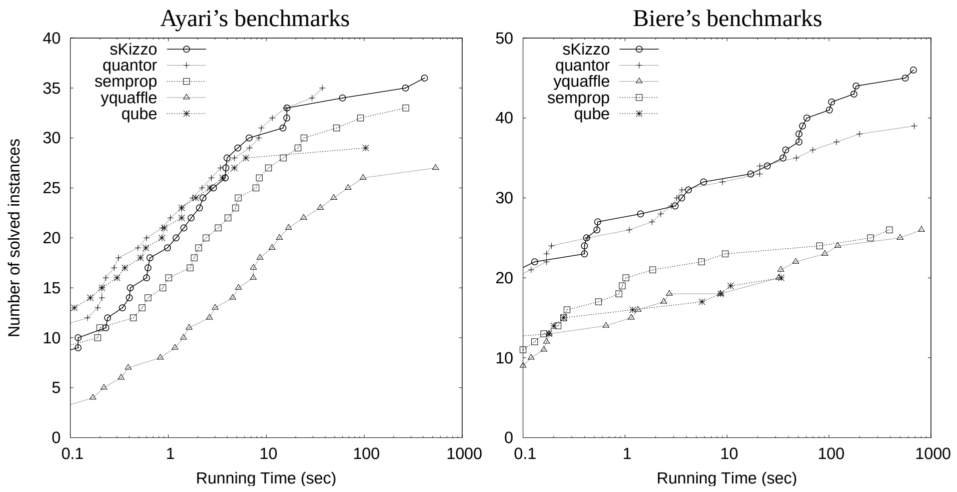


Fig. 1. Comparison with other solvers over two groups of families

for modern solvers, and some of them have never been solved), and (3) Biere's benchmarks [4], made up of 64 instances divided into 4 families, where the n-th instance in each family refers to model checking problem on a n-bit counter. The verification is easy for BDD-based symbolic MC and very difficult for SAT-based bounded MC, as it captures the worst-case scenario in which the number of steps necessary to falsify the property equals the diameter of the system. QBF reasoning has been shown not to outperform SAT-based reasoning (Bounded Model Checking) on these benchmarks.

Table 1 measures the relative importance of symbolic reasoning w.r.t. all the other reasoning strategies. It puts side by side the symbolic/ground size of a few instances before and after $Norm_{\mathcal{R}}$ is applied for the first time. The last column gives the amount of time spent in (the first application of) symbolic reasoning. When this percentage is equal to 100%, the instance is just symbolically solved. As expected, the ground size of instances is always reduced, whilst the symbolic size of some of them is increased as an effect of symbolic reasoning. The reduction ratio for the ground size is quite family-dependent, though not sensibly instance-depending. Most of the simpler families are completely solved by symbolic reasoning. Conversely, for more complex instances symbolic reasoning does not suffices. Quite often, the number of ground clauses before symbolic reasoning is intractable (state-of-the-art solvers can afford millions clauses, not billions). Some of them stays unaffordable even after $Norm_{\mathcal{R}}$, but many undergo a strong reduction of the ground size. Several problems exist that—thought not strongly reduced during the first call to $Norm_{\mathcal{R}}$—are hardly simplified during the recursive calls (not shown in the table). The overall effect of symbolic reasoning is quite incisive.

Figure 1 compares sKizzo with publically available state-of-the-art solvers, among which we find the three top-rated solvers according to most of the results presented in [20] (see also Section 6). The number of solved instances in two groups of families is plotted against the (non-cumulative) time taken to solve such instances. The overall performance is quite impressive, especially if we take into consideration that sKizzo is just a first non-optimized implementation.

6 Related work and discussion

Most QBF solvers leverage revised versions of search-based techniques developed in the SAT framework, ranging from the extension of resolution-based reasoning [18] to the employment of lookback techniques [23], encountering along the way a key contribution by Cadoli, Giovanardi and Schaerf [7] in which the original extension of DPLL to QBF is presented. Up to a certain point, these extensions have been successful. In the solver evaluation reported in [21], all the competitive solvers (such as QSAT [30], QSOLVE [11], QUAFFLE [34], QuBE [15], SEMPROP [22]) are search-based.

A few alternative algorithms for QBF are emerging [20]. Some of them reverse the order in which quantifiers are considered (such as Quantor [4]), others employ some compact representation for the problem (such as ZQSAT [14] and QMRES/QBDD [28]). Many restate the very goal of the solver: it is no longer a matter of *searching for a solution*, rather an attempt to directly *solve the instance* (this distinction traces back to [9, 10]). Resolution-based solving techniques have also received renewed attention, especially when used in conjunction with compressed representation for clauses [8, 13, 25]. In the SAT framework, these so-called *symbolic* approaches show a certain strength on specific classes of instances, but seem to be not competitive in general [27]. In the QBF scenario, both the idea of *compressed/symbolic* representations, and the shift from *searching* to *solving* are more promising [28, 4, 14].

The foundational work of Skolem [31] has had the widest possible application. We here just cite a recent work by Jackson [16]. Forms of reasoning about binary subformulas are regarded as an effective pre-processing step in the propositional framework [2]. The interest in binary decision diagrams as a tool for manipulating boolean functions traces back to the seminal work by Bryant [6]. Their usage in SAT/QBF satisfiability algorithms have been explored at least in [33, 8, 25, 13, 27, 28, 14].

Several features distinguish our approach from previous ones. For example, it: (1) largely abstracts over variable ordering and number of alternations in the prefix; (2) explicitly leverages skolemization; (3) profits from the peculiar structure of QBF-derived instances to symbolically represent them; (4) advantageously integrates search-based and solving decision strategies in QBF reasoning; (5) repeatedly engages a SAT solver as an oracle; (6) employs a hybrid PROP/QBF branching style. For further differences and an in-depth comparison, see [3].

7 Conclusions and future work

Our work is motivated by the outstanding potential of QBF in applications. Advances in decision procedures for this formalism are ardently expected, and quantified reasoners worhty of inheriting the amazing success of SAT solvers are a looming possibility. In this respect, we firstly succeed to efficiently retain both the expressive power of quantification and the strength of the purely propositional reasoning. Our preliminary experimental evaluation yields remarkable results. Large room for improvements exist as (1) our implementation is just a first, non-optimized prototype, and (2) several effective QBF and SAT reasoning techniques (q-resolution, subsumption control, backjumping, etc.) have been left out of the first implementation to focus on the main topic.

To further investigate our guideline, we are (1) strengthening the symbolic machinery by adding new rules, (2) conceiving a symbolic *model verifier*, and (3) designing the integration with an industrial-scale model checker.

Acknowledgements

I thank Gigina Aiello and Paolo Traverso for supporting my research efforts, and Sara Bernardini for the many days she has spent on listening to my early ideas on symbolic skolemization. I'm grateful to Marco Cadoli for discussing this work with me. Finally, Amedeo Cesta deserves a special thought for his indefatigable encouragement.

References

1. A. Ayari and D. Basin. Bounded Model Construction for Monadic Second-order Logics. In *Proc. of CAV'00*, 2000.
2. F. Bacchus and J. Winter. Effective Preprocessing with Hyper-Resolution and Equality Reduction. In *Proc. of SAT'03*, 2003.
3. M. Benedetti. sKizzo: a QBF Decision Procedure based on Propositional Skolemization and Symbolic Reasoning, Tech.Rep. 04-11-03, ITC-irst, available at `sra.itc.it/people/benedetti/sKizzo`, 2004.
4. A. Biere. Resolve and Expand. In *Proc. of SAT'04*, pages 238–246, 2004.
5. A. Biere, A. Cimatti, E. M. Clarke, M. Fujita, and Y. Zhu. Symbolic Model Checking without BDDs. In *Proc. of Design Automation Conference*, volume 1579, pages 193–207, 1999.
6. R. E. Bryant. Graph-based algorithms for Boolean function manipulation. *IEEE Transaction on Computing*, C-35(8):677–691, 1986.
7. Marco Cadoli, Andrea Giovanardi, and Marco Schaerf. An algorithm to evaluate quantified boolean formulae. In *Proceedings of the fifteenth national/tenth conference on Artificial intelligence/Innovative applications of artificial intelligence*, pages 262–267. American Association for Artificial Intelligence, 1998.
8. P. Chatalic and L. Simon. Multi-Resolution on compressed sets of clauses. In *Proceedings of the Twelfth International Conference on Tools with Artificial Intelligence (ICTAI'00)*, 2000.
9. M. Davis, G. Logemann, and D. Loveland. A machine program for theorem proving. *Journal of the ACM*, 5:394–397, 1962.
10. M. Davis and H. Putnam. A computing procedure for quantification theory. *Journal of the ACM*, 7, 1960.
11. R. Feldmann, B. Monien, and S. Schamberger. A Distributed Algorithm to Evaluate Quantified Boolean Formulas. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, pages 285–290, 2000.
12. M. Fitting. *First-Order Logic and Automated Theorem Proving*. Springer Verlag, 1996.
13. J. Franco, M. Kouril, J. Schlipf, J. Ward, S. Weaver, M. Dransfield, and W. Vanfleet. SBSAT: a state-based, BDD-based satisfiability solver. In *Proceedings of SAT'03*, 2003.
14. M. GhasemZadeh, V. Klotz, and C. Meinel. ZQSAT: A QSAT Solver based on Zero-suppressed Binary Decision Diagrams, available at `www.informatik.uni-trier.de/TI/bdd-research/zqsat/zqsat.html`, 2004.
15. E. Giunchiglia, M. Narizzano, and A. Tacchella. QuBE: A system for deciding Quantified Boolean Formulas Satisfiability. In *Proc. of the International Joint Conference on Automated Reasoning (IJCAR'2001)*, 2001.
16. Daniel Jackson. Automating first-order relational logic. In *Proceedings of the 8th ACM SIGSOFT international symposium on Foundations of software engineering*, pages 130–139. ACM Press, 2000.
17. H. Kautz and B. Selman. Planning as satisfiability. In *Proc. of ECAI 1992*, pages 359–363.
18. H. Kleine-Buning, M. Karpinski, and A. Flogel. Resolution for quantified Boolean formulas. *Information and Computation*, 117(1):12–18, 1995.
19. T. Larrabee. Test pattern generation using boolean satisfiability. In *IEEE Transaction on Computer-aided Design*, pages 4–15, 1992.
20. D. Le Berre, M. Narizzano, L. Simon, and A. Tacchella. Second QBF solvers evaluation, avaliable on-line at `www.qbflib.org`, 2004.
21. D. Le Berre, L. Simon, and A. Tacchella. Challenges in the QBF arena: the SAT'03 evaluation of QBF solvers, avaliable on-line at `www.qbflib.org`, 2003.
22. R. Letz. Advances in Decision Procedures for Quantified Boolean Formulas. In *Proceedings of the First International Workshop on Quantified Boolean Formulae (QBF'01)*, pages 55–64, 2001.
23. R. Letz. Lemma and model caching in decision procedures for quantified boolean formulas. In *Proc. of the Int. Conf. on Automated Reasoning with Analytic Tableaux and Related Methods*, pages 160–175. Springer-Verlag, 2002.
24. M. W. Moskewicz, C. F. Madigan, Y. Zhao, L. Zhang, and S. Malik. Chaff: Engineering an Efficient SAT Solver. In *proceedings of the 38th Design Automation Conference*, 2001.
25. D. B. Motter and I. L. Markov. A compressed, breadth-first search for satisfiability. *LNCS*, 2409:29–42, 2002.
26. A. Nonnengart and C. Weidenbach. Computing Small Clause Normal Forms. In Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning*, chapter 6, pages 335 – 367. Elsevier, Amsterdam, Netherlands, 2001.
27. G. Pan and M.Y. Vardi. Search vs. Symbolic Techniques in Satisfiability Solving. In *Proceedings of SAT 2004*, 2004.
28. G. Pan and M.Y. Vardi. Symbolic Decision Procedures for QBF. In *Proceedings of the Tenth International Conference on Principles and Practice of Constraint Programming (CP04)*, 2004.
29. J. Rintanen. Construction Conditional Plans by a Theorem-prover. *Journal of A. I. Research*, pages 323–352, 1999.
30. 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.
31. T. Skolem. Logico-combinatorial investigations in the satisfiability or provability of mathematical propositions: a simplified proof of a theorem by L. Löwenheim and generalizations of the theorem. In *From Frege to Gödel. A Source Book in Mathematical Logic, 1879-1931*, pages 252–263. Harvard University Press, Cambridge, 1967 (1920).
32. Fabio Somenzi. Colorado University Binary Decision Diagrams, `vlsi.colorado.edu/~fabio/CUDD`, 1995.
33. T. E. Uribe and M. E. Stickel. Ordered binary decision diagrams and the Davis-Putnam procedure. In J. P. Jouannaud, editor, *1st International Conference on Constraints in Computational Logics*, volume 845, pages 34–49, 1994.
34. L. Zhang and S. Malik. Towards Symmetric Treatment of Conflicts And Satisfaction in Quantified Boolean Satisfiability Solver. In *Proc. of CP'02*, 2002.