Expansion-Based QBF Solving Without Recursion

Roderick Bloem, Nicolas Braud-Santoni, Vedad Hadzic, TU Graz
Uwe Egly, Florian Lonsing, TU Wien
Martina Seidl, JKU Linz

***Abstract*—In recent years, expansion-based techniques have been shown to be very powerful in theory and practice for solving quantified Boolean formulas (QBF), the extension of propositional formulas with existential and universal quantifiers over Boolean variables. Such approaches partially expand one type of variable (either existential or universal) and pass the obtained formula to a SAT solver for deciding the QBF. State-of-the-art expansion-based solvers process the given formula quantifier-block wise and recursively apply expansion until a solution is found.**

In this paper, we present a novel algorithm for expansion-based QBF solving that deals with the whole quantifier prefix at once. Hence recursive applications of the expansion principle are avoided. Experiments indicate that the performance of our simple approach is comparable with the state of the art of QBF solving, especially in combination with other solving techniques.

I. Introduction

Efficient tools for deciding the satisfiability of Boolean formulas (SAT solvers) are the core technology in many verification and synthesis approaches [45]. However, verification and synthesis problems are often beyond the complexity class NP as captured by SAT, requiring more powerful formalisms like *quantified Boolean formulas* (QBFs). QBFs extend propositional formulas by universal and existential quantifiers over Boolean variables [32] resulting in a decision problem that is PSPACE-complete. Applications from verification and synthesis [8], [13], [14], [18], [20], [24], realizability checking [19], bounded model checking [16], [48], and planning [17], [41] motivate the quest for efficient QBF solvers.

Unlike for SAT, where *conflict-driven clause learning* (CDCL) is the single dominant solving approach for practical problems, two dominant approaches exist for QBF solving. On one hand, CDCL has been successfully extended to QCDCL that enables clause and cube learning [21], [35], [47]. On the other hand, *variable expansion* has become very popular. In short, expansion-based solvers eliminate one kind of variables by assigning them truth values and solve the resulting propositional formula with a SAT solver. For QBFs with one quantifier alternation (2QBF), a natural approach is to use two SAT solvers: one that deals with the existentially quantified variables and another one that deals with the universally quantified variables. For generalising this SAT-based approach to QBFs with an arbitrary number of quantifier alternations, expansion is recursively applied per quantifier block, requiring multiple SAT solvers. As noted by Rabe and Tentrup [39], these CEGAR-based approaches show poor performance for formulas with many quantifier alternations in general.

This work has been supported by the Austrian Science Fund (FWF) under projects W1255-N23, S11406-N23, S11408-N23, and S11409-N23.

In this paper, we present a novel solving algorithm based on non-recursive expansion for QBFs with arbitrary quantifier prefixes using only two SAT solvers. Our approach of non-recursive expansion is theoretically (i.e., from a proof complexity perspective) equivalent to approaches that apply recursive expansion since both non-recursive and recursive expansion rely on the ∀Exp+Res proof system [5]. However, the non-recursive expansion has practical implications such as a modified search strategy. That is, the use of recursive or non-recursive expansion results in different search strategies for the proof. With respect to proof search, there is an analogy to, e.g., implementations of resolution-based CDCL SAT solvers that employ different search heuristics.

In addition, we implemented a hybrid approach that combines clause learning with non-recursive expansion-based solving for exploiting the power of QCDCL. Our experiments indicate that this hybrid approach performs very well, especially on formulas with multiple quantifier alternations.

This paper is structured as follows. After a review of related work in the next section, we introduce the necessary preliminaries in Section III. After a short recapitulation of expansion in Section IV, our novel non-recursive expansion-based algorithm is presented in Section V. Implementation details are discussed in Section VI together with a short discussion of the hybrid approach. In Section VII we compare our approach to state-of-the-art solvers.

II. Related Work

Already the early QBF solvers Qubos [2] and Quantor [2] incorporate selective quantifier expansion for eliminating one kind of quantification to reduce the given QBF to a propositional formula. The resulting propositional formula is then solved by calling a SAT solver once. Qubos and Quantor impressively demonstrated the power of expanding universal variables but also showed its enormous memory consumption. As a pragmatic compromise, bounded universal expansion was introduced for efficient preprocessing [11], [22], [23], [46].

The first approach which uses two alternate SAT solvers A and B for solving 2QBF, i.e., QBFs of the form $\forall U \exists E.\phi$, was presented in [40]. Solver A is initialised with ϕ, B with the empty formula. Both propositional formulas are incrementally refined with satisfying assignments found by the other solver. If A finds its formula unsatisfiable, then the QBF is false. Otherwise, the negation of the universal part of the satisfying assignment is passed to solver B. If solver B finds its formula unsatisfiable, then the QBF is true. Otherwise, the existential part of the satisfying assignment is passed to solver A. Janota and Marques-Silva generalised the idea of alternating SAT

solvers [31] such that one solver deals with the existentially quantified variables and one solver deals with the universally quantified variables exclusively. Solver A gets *instantiations* of ϕ in which the universal variables are assigned, and solver B gets instantiations of $\neg\phi$ in which the existential variables are assigned. The satisfying assignment found by one solver is used to obtain a new instantiation for the other. This loop is repeated until one solver returns unsatisfiable. This approach realises a natural application of the counter-example guided abstraction refinement (CEGAR) paradigm [15]. A detailed survey on 2QBF solving is given in [3].

A significant advancement of expansion-based solving for QBF with an arbitrary number of quantifier alternations was made with the solver RAReQS [26], [27], which recursively applies the previously discussed 2QBF approach [31] for each quantifier alternation. The approach turned out to be highly competitive.[1] For formalising this solving approach the calculus ∀Exp+Res was introduced [5], and proof-theoretical investigations revealed the orthogonal strength of ∀Exp+Res and Q-resolution [33], the QBF variant of the resolution calculus that forms the basis for QCDCL-based solvers. Research on the proof complexity of QBF has identified an exponential separation between Q-resolution and the ∀Exp+Res system. There are families of QBFs for which any Q-resolution proof has exponential size, in contrast to ∀Exp+Res proofs of polynomial size, and vice versa. Hence these two systems have orthogonal strength.

Recent work successfully combines machine learning with this CEGAR approach [25]. Motivated by the success of expansion-based QBF solving, several other approaches [10], [30], [39], [42]–[44] have been presented that are based on levelised SAT solving, i.e., one SAT solver is responsible for the variables of one quantifier block. In this paper, we also introduce a solving approach that is based upon propositional abstraction but considers the whole quantifier prefix at once.

III. Preliminaries

The QBFs considered in this paper are in prenex normal form $\Pi.\phi$ where Π is a quantifier prefix $Q_1x_1Q_2x_2\dots Q_nx_n$ over the set of variables $X = \{x_1, \dots, x_n\}$ with $Q_i \in \{\forall, \exists\}$ and $x_i \neq x_j$ for $i \neq j$. The propositional formula ϕ contains only variables from X. Unless stated otherwise, we do not make any assumptions on the structure of ϕ. Sometimes $\Pi.\phi$ is in *prenex conjunctive normal form* (PCNF), i.e., Π is a prefix as introduced before and ϕ is a conjunction of clauses. A clause is a disjunction of literals, and a literal is a variable or the negation of a variable. The prefix imposes the order $<_\Pi$ on the elements of X such that $x_i <_\Pi x_j$ if $i < j$. By U_Π (E_Π) we denote the set of universally (existentially) quantified variables of the prefix Π. If clear from the context we omit the subscript Π. We assume the standard semantics of QBF. A QBF consisting of only the syntactic truth constant $\bot$ ($\top$) is false (true). A QBF $\forall x\Pi.\phi$ is true if $\Pi.\phi[x \leftarrow \top]$ and $\Pi.\phi[x \leftarrow \bot]$ are both true, where $\phi[x \leftarrow t]$ is the substitution of x by t in ϕ. A QBF $\exists x\Pi.\phi$ is true if $\Pi.\phi[x \leftarrow \top]$ or $\Pi.\phi[x \leftarrow \bot]$ is true.

[1] http://www.qbflib.org

Given a set of variables X, we call a function $\sigma\colon X \to \{\top, \bot, \epsilon\}$ an assignment for X. If there is an $x \in X$ with $\sigma(x) = \epsilon$ then σ is a *partial* assignment, otherwise σ is a *full* assignment of X. Informally, $\sigma(x) = \epsilon$ means that σ does not assign a truth value to variable x. A restriction $\sigma|_Y\colon Y \to \{\top, \bot, \epsilon\}$ of assignment $\sigma\colon X \to \{\top, \bot, \epsilon\}$ to $Y \subseteq X$ is defined by $\sigma|_Y(x) = \sigma(x)$ if $x \in Y$, otherwise $\sigma|_Y(x) = \epsilon$. By Σ_X we denote the set of all full assignments $\sigma\colon X \to \{\top, \bot\}$. Let ϕ be a propositional formula over X. By $\sigma(\phi)$ we denote the application of assignment $\sigma\colon X \to \{\top, \bot, \epsilon\}$ on ϕ, i.e., $\sigma(\phi)$ is the formula obtained by replacing variables $x \in X$ by $\sigma(x)$ if $\sigma(x) \in \{\top, \bot\}$ and performing standard propositional simplifications. Let ϕ, ψ be propositional formulas over the set of variables X. If for every full assignment $\sigma \in \Sigma_X$, $\sigma(\phi) = \sigma(\psi)$ then ϕ and ψ are equivalent. Let $\tau\colon X \to \{\top, \bot, \epsilon\}$ and $\sigma\colon Y \to \{\top, \bot, \epsilon\}$ be assignments such that for every $x \in X \cap Y$, $\tau(x) = \sigma(x)$ if $\tau(x) \neq \epsilon$ and $\sigma(x) \neq \epsilon$. Then the composite assignment of σ and τ is denoted by $\sigma\tau\colon X \cup Y \to \{\top, \bot, \epsilon\}$ and for every propositional formula ϕ over $X \cup Y$, it holds that $\sigma\tau(\phi) = \tau\sigma(\phi) = \sigma(\tau(\phi)) = \tau(\sigma(\phi))$. Furthermore, $\sigma\sigma = \sigma$ for any assignment σ.

Example 1. *Let $\sigma\colon X \to \{\top, \bot, \epsilon\}$ be an assignment over variables $\{a, b, x, y\}$ defined by $\sigma(a) = \top$, $\sigma(b) = \epsilon$, $\sigma(x) = \top$, and $\sigma(y) = \epsilon$. The restriction $\tau = \sigma|_Y$ of σ to $Y = \{x, y\}$ is given by $\tau(a) = \epsilon$, $\tau(b) = \epsilon$, $\tau(x) = \top$, $\tau(y) = \epsilon$. For the propositional formula $\phi = (x \vee a \vee y) \wedge (\neg x \vee \neg a \vee y) \wedge (\neg y \vee b)$, the application of σ and τ on ϕ gives us $\sigma(\phi) = y \wedge (\neg y \vee b)$ and $\tau(\phi) = (\neg a \vee y) \wedge (\neg y \vee b)$.*

IV. Expansion

In the following, we introduce the notation and terminology used for describing expansion-based QBF solving in general, and the algorithm introduced in the next section in particular. We first define the notion of *instantiation* that is inspired by the axiom rule of the calculus ∀Exp+Res [29].

Definition 1. *Let $\Pi.\phi$ be a QBF with prefix $\Pi = Q_1x_1 \dots Q_nx_n$ over the set of variables $X = \{x_1, \dots, x_n\}$ and $\sigma\colon Y \to \{\top, \bot, \epsilon\}$ with $Y \subseteq X$ an assignment. If $Y \subset X$, we extend the domain of σ to X by setting $\sigma(x) = \epsilon$ if $x \notin Y$. The* instantiation of ϕ by *σ, denoted by ϕ^σ, is obtained from ϕ as follows:*

1) *all variables $x \in X$ with $\sigma(x) \neq \epsilon$ are set to $\sigma(x)$;*
2) *all variables $x \in X$ with $\sigma(x) = \epsilon$ are replaced by x^ω where annotation ω is uniquely defined by the sequence $\sigma(x_{k_1})\sigma(x_{k_2}) \dots \sigma(x_{k_m})$ such that the set formed from the variables x_{k_i} contains all variables of X with $x_{k_i} <_\Pi x$. Furthermore, $x_{k_i} <_\Pi x_{k_j}$ if $k_i < k_j$.*

If we instantiate a QBF $\Pi.\phi$ with the full assignment $\sigma\colon U_\Pi \to \{\top, \bot\}$ of the universal variables, we obtain a propositional formula that contains only (possibly annotated) variables from E_Π. The dual holds for the instantiation by a full assignment $\sigma\colon E_\Pi \to \{\top, \bot\}$.

Example 2. *Given the QBF $\forall a \exists x \forall b \exists y.\phi$ with $\phi = ((x \vee a \vee y) \wedge (\neg x \vee \neg a \vee y) \wedge (\neg y \vee b))$. Then $U = \{a, b\}$ and $E = \{x, y\}$. Let $\sigma\colon U \to \{\top, \bot, \epsilon\}$ be defined by $\sigma(a) = \top$*

and $\sigma(b) = \bot$. Then $\phi^\sigma = (\neg x^\top \vee y^{\top\bot}) \wedge \neg y^{\top\bot}$. Further, let $\tau\colon E \to \{\top, \bot, \epsilon\}$ with $\tau(x) = \bot$ and $\tau(y) = \bot$. Then $\phi^\tau = a$. Note that a is not annotated because it occurs in the first quantifier block.

Sometimes we want to remove the annotations again from an assignment or an instantiated formula. Therefore, we introduce the following notation. Let ϕ^σ be an instantiation by assignment $\sigma\colon X \to \{\top, \bot, \epsilon\}$ and X^σ the set of annotated variables. If we have an assignment $\tau\colon X^\sigma \to \{\top, \bot, \epsilon\}$, then we define $\tau^{-\sigma}\colon X \to \{\top, \bot\}$ by $\tau^{-\sigma}(x) = \tau(x^\sigma)$ for $x^\sigma \in X^\sigma$. If we have an instantiated formula ϕ^σ, the $(\phi^\sigma)^{-\sigma}$ is the formula obtained by replacing every annotated variable $x^\sigma \in X^\sigma$ by x. In general, $(\phi^\sigma)^{-\sigma} \neq \phi$.

Lemma 1. *Let $\Pi.\phi$ be a QBF with variables X and $\sigma\colon X \to \{\top, \bot, \epsilon\}$ be a partial assignment. Then $(\phi^\sigma)^{-\sigma}$ and $\sigma(\phi)$ are equivalent.*

Proof. By induction over the formula structure. For the base case let $\phi = x$ with $x \in X$. If $\sigma(x) = \epsilon$, $\sigma(\phi) = x$ and $\phi^\sigma = x$. Then $(\phi^\sigma)^{-\sigma} = x$. Otherwise, $\phi^\sigma = \sigma(x)$. Obviously, $\sigma(\phi) = \sigma(x) = (\sigma(x))^{-\sigma} \in \{\top, \bot\}$. The induction step naturally follows from the semantics of the logical connectives. □

Example 3. *Reconsider the propositional formula ϕ and assignments σ, τ from above (Example 2). Then $(\phi^\sigma)^{-\sigma} = ((\neg x^\top \vee y^{\top\bot}) \wedge \neg y^{\top\bot})^{-\sigma} = (\neg x \vee y) \wedge \neg y$. Furthermore, $(\phi^\tau)^{-\tau} = (a)^{-\tau} = a$.*

Finally, we specify the semantics of a QBF in terms of universal and existential expansion on which expansion-based QBF solving is founded.

Lemma 2. *Let $\Phi = \Pi.\phi$ be a QBF with universal variables U. There is a set of assignments $A \subseteq \Sigma_U$ with $\bigwedge_{\alpha\in A} \phi^\alpha$ is unsatisfiable if and only if Φ is false.*

The lemma above has a dual version for true QBFs. This duality plays a prominent role in our novel solving algorithm.

Lemma 3. *Let $\Phi = \Pi.\phi$ be a QBF with existential variables E. There is a set of assignments $S \subseteq \Sigma_E$ with $\bigvee_{\sigma\in S} \phi^\sigma$ is valid if and only if Φ is true.*

V. A Non-Recursive Algorithm for Expansion-Based QBF Solving

The pseudo-code in Figure 1 summarises the basic idea of our novel approach for solving the QBF $\Pi.\phi$ with universal variables U and existential variables E.

First, an arbitrary assignment α_0 for the universal variables is selected in Line 1. The instantiation ϕ^{α_0} is handed over to a SAT solver. If ϕ^{α_0} is unsatisfiable, then $\Pi.\phi$ is false and the algorithm returns. Otherwise, $\tau\colon E^{\alpha_0} \to \{\top, \bot\}$ is a satisfying assignment of ϕ^{α_0}. Let σ_1 denote the assignment $\tau^{-\alpha_0}$. Then $\alpha_0\sigma_1$ is a satisfying assignment of ϕ.

Next, the propositional formula $\neg\phi^{\sigma_1}$ is handed over to a SAT solver for checking the validity of ϕ^{σ_1}. If $\neg\phi^{\sigma_1}$ is unsatisfiable, then $\Pi.\phi$ is true and the algorithm returns. If $\neg\phi^{\sigma_1}$ is satisfiable, then $\rho\colon U^{\sigma_1} \to \{\top, \bot\}$ is a satisfying assignment of $\neg\phi^{\sigma_1}$. Let α_1 denote the assignment $\rho^{-\sigma_1}$. Then $\alpha_1\sigma_1$ is a satisfying assignment for $\neg\phi$. The following lemma shows that α_0 and α_1 are different.

input : QBF $\Pi.\phi$ with universal variables U and existential variables E
output: truth value of $\Pi.\phi$

1 $A_0 := \{\alpha_0\}$, where $\alpha_0\colon U \to \{\top, \bot\}$ is an arbitrary assignment
2 $S_0 := \emptyset$
3 $i := 1$
4 **while** *true* **do**
5 $\quad (\mathit{isUnsat}, \tau) := \mathrm{SAT}(\bigwedge_{\alpha\in A_{i-1}} \phi^\alpha)$
6 $\quad$ **if** *isUnsat* **then** return false;
7 $\quad S_i := S_{i-1} \cup \{(\tau|_{E^\alpha})^{-\alpha} \mid \alpha \in A_{i-1}\}$
8 $\quad (\mathit{isUnsat}, \rho) := \mathrm{SAT}(\bigwedge_{\sigma\in S_i} \neg\phi^\sigma)$
9 $\quad$ **if** *isUnsat* **then** return true;
10 $\quad A_i := A_{i-1} \cup \{(\rho|_{U^\sigma})^{-\sigma} \mid \sigma \in S_i\}$
11 $\quad i$++
12 **end**

Figure 1: Non-Recursive Expansion-Based Algorithm

Lemma 4. *Let $\Pi.\phi$ be a QBF with universal variables U and existential variables E. Further, let $\alpha\colon U \to \{\top, \bot\}$ be an assignment such that the instantiation ϕ^α is satisfiable and has the satisfying assignment $\tau\colon E^\alpha \to \{\top, \bot\}$. Let $\sigma\colon E \to \{\top, \bot\}$ with $\sigma = \tau^{-\alpha}$. Then α falsifies $(\neg\phi^\sigma)^{-\sigma}$.*

Proof. Since ϕ^α is satisfied by τ, ϕ is satisfied by the composite assignment $\alpha\tau^{-\alpha} = \alpha\sigma$, and therefore $\neg\phi$ is falsified by $\alpha\sigma$. Then α falsifies $\sigma(\neg\phi)$. According to Lemma 1 $\sigma(\neg\phi)$ is equivalent to $(\neg\phi^\sigma)^{-\sigma}$. Then α also falsifies $(\neg\phi^\sigma)^{-\sigma}$. □

In the next round of the algorithm, the propositional formula $\phi^{\alpha_0} \wedge \phi^{\alpha_1}$ is handed over to a SAT solver. If this formula is unsatisfiable, $\Pi.\phi$ is false and the algorithm returns. Otherwise, it is satisfiable under some assignment $\tau\colon E^{\alpha_0} \cup E^{\alpha_1} \to \{\top, \bot\}$, then at least one new assignment $\sigma_2\colon E \to \{\top, \bot\}$ with $\sigma_2 \neq \sigma_1$ can be extracted from $\tau|_{E^{\alpha_i}}$ with $0 \leq i \leq 1$. This assignment is then used for obtaining a new propositional formula $\phi^{\sigma_1} \vee \phi^{\sigma_2}$. To show the validity of this formula, its negation is passed to a SAT solver. If this formula is unsatisfiable, $\Pi.\phi$ is true and the algorithm returns. Otherwise, it is satisfiable under the assignment $\rho\colon U^{\sigma_1} \cup U^{\sigma_2} \to \{\top, \bot\}$. A new assignment $\alpha_2\colon U \to \{\top, \bot\}$ with $\alpha_2 \neq \alpha_1 \neq \alpha_0$ is obtained from $\rho|_{A^{\sigma_i}}$ with $1 \leq i \leq 2$. This assignment is then used in the next round of the algorithm. In this way, the propositional formulas $\bigwedge_{\alpha\in\Sigma_U} \phi^\alpha$ and $\bigvee_{\sigma\in\Sigma_E} \phi^\sigma$ are generated. If $\bigwedge_{\alpha\in A} \phi^\alpha$ is unsatisfiable for some $A \subseteq \Sigma_U$, by Lemma 2 $\Pi.\phi$ is false. Dually, if $\bigvee_{\sigma\in S} \phi^\sigma$ is valid for some $S \subseteq \Sigma_E$, by Lemma 3 $\Pi.\phi$ is true. The algorithm iteratively extends the sets A and S by adding parts of satisfying assignments of ϕ to S and parts of falsifying assignments to A. In particular, A is extended by assignments of the universal variables and S is extended by assignments of the existential variables. The order

in which assignments are considered depends on the used SAT solver.

Example 4. *We show how to solve the QBF $\forall a \exists x \forall b \exists y.\phi$ with $E = \{x, y\}$, $U = \{a, b\}$, and $\phi = ((a \vee x \vee y) \wedge (\neg a \vee \neg x \vee y) \wedge (b \vee \neg y))$ with the algorithm presented above. This formula can be solved in two iterations:*

Init: We start with some random assignment $\alpha_0 \colon U \to \{\top, \bot\}$, for example with $\alpha_0(a) = \top$ and $\alpha_0(b) = \bot$.

Iteration 1: The formula $\phi^{\alpha_0} = (\neg x^{\top} \vee y^{\top\bot}) \wedge \neg y^{\top\bot}$ is passed to a SAT solver and found satisfiable under the assignment $\tau \colon E^{\alpha_0} \to \{\top, \bot\}$ with $\tau(x^{\top}) = \bot$ and $\tau(y^{\top\bot}) = \bot$. By removing the variable annotations we get assignment $\sigma_1 = (\tau|_{E^{\alpha_0}})^{-\alpha_0}$, where $\sigma_1 \colon E \to \{\top, \bot\}$ with $\sigma_1(x) = \bot$ and $\sigma_1(y) = \bot$. Based on this assignment we obtain $\phi^{\sigma_1} = a$. The formula $\neg\phi^{\sigma_1}$ is passed to a SAT solver. It is satisfiable and has the satisfying assignment $\rho \colon U^{\sigma_1} \to \{\top, \bot\}$ with $\rho(a) = \bot$ and $\rho(b^{\bot}) = \top$, which we then reduce to $\alpha_1 = (\rho|_{U^{\sigma_1}})^{-\sigma_1}$ where $\alpha_1 \colon U \to \{\top, \bot\}$ with $\alpha_1(a) = \bot$ and $\alpha_1(b) = \top$.

Iteration 2: The formula $\phi^{\alpha_0} \wedge \phi^{\alpha_1} = (\neg x^{\top} \vee y^{\top\bot}) \wedge \neg y^{\top\bot} \wedge (x^{\bot} \vee y^{\bot\top})$ is passed to a SAT solver in the second iteration. It is satisfiable and one satisfying assignment is $\tau \colon E^{\alpha_0} \cup E^{\alpha_1} \to \{\top, \bot\}$ with $\tau(x^{\top}) = \bot, \tau(x^{\bot}) = \top, \tau(y^{\top\bot}) = \bot, \tau(y^{\bot\top}) = \bot$. From τ, we can extract the assignment $\sigma_2 = (\tau|_{E^{\alpha_1}})^{-\alpha_1}$ where $\sigma_2 \colon E \to \{\top, \bot\}$ with $\sigma_2(x) = \top$ and $\sigma_2(y) = \bot$. Note that for any choice of τ, $\sigma_2 \neq \sigma_1$. Next, we construct $\phi^{\sigma_1} \vee \phi^{\sigma_2} = a \vee \neg a$. This formula is a tautology, so its negation that is passed to a SAT solver is unsatisfiable, hence $\Pi.\phi$ is true.

The soundness of our algorithm immediately follows from Lemmas 2 and 3: the algorithm returns false (true) if, in some iteration i, it finds that the current partial expansion $\bigwedge_{\alpha \in A_{i-1}} \phi^{\alpha}$ (respectively $\bigwedge_{\sigma \in S_i} \neg\phi^{\sigma}$) is unsatisfiable.

Theorem 1. *The algorithm shown in Figure 1 is sound.*

For showing that the algorithm also terminates, we argue that sets A_i and S_i increase in iteration $i+1$. To this end, we have to relate the variables of the QBF, the annotated variables as well as their assignments. Before we give the proof, we first consider another example in which we illustrate how the different assignments are related.

Example 5. *We show one possible run of the algorithm presented above for the QBF $\Phi := \forall a \exists x \forall b \exists y.\phi$ with*

$$\phi := (a \wedge b \wedge \neg x \wedge \neg y) \vee (\neg a \wedge x \wedge (b \leftrightarrow y))$$

and how it iteratively generates the sets Σ_U and Σ_E. Figure 2 shows the expansion trees that are implicitly built during the search. An expansion tree relates the variables of the partial expansion of Φ constructed from A_i (left column) and S_i (right column). Solid edges indicate that the variable on the top has been set by an assignment from A_i or S_i, and dotted edges indicate that the variable has to be assigned a value by the SAT solver. The order of the (annotated) variables in the expansion tree respects the order of the (original) variables in the prefix.

Init: For the initialisation of A_0, an arbitrary assignment $\alpha_0 \colon U \to \{\top, \bot\}$ is chosen. Let $\alpha_0(a) = \bot$ and $\alpha_0(b) = \bot$.

Iteration 1: $\phi^{\alpha_0} := x^{\bot} \wedge \neg y^{\bot\bot}$ is satisfiable. Assignment $\sigma_1 \colon E \to \{\top, \bot\}$, with $\sigma_1(x) = \top$ and $\sigma_1(y) = \bot$, is extracted from model $\tau \colon E^{\alpha_1} \to \{\top, \bot\}$ and added to S_1. Now $\phi^{\sigma_1} := \neg a \wedge \neg b^{\top}$ is checked for validity. Assignment $\alpha_1 \colon U \to \{\top, \bot\}$, with $\alpha_1(a) = \bot$ and $\alpha_1(b) = \top$, obtained from counter-example $\rho \colon U^{\sigma_1} \to \{\top, \bot\}$ is added to A_1.

Iteration 2: Next, $\phi^{\alpha_0} \wedge \phi^{\alpha_1}$ with $\phi^{\alpha_1} := x^{\bot} \wedge y^{\bot\top}$ is checked. From model $\tau \colon E^{\alpha_0} \cup E^{\alpha_1} \to \{\top, \bot\}$, again σ_1 can be extracted for ϕ^{α_0}. For ϕ^{α_1} a new assignment σ_2 which is not in S_1 is found and added to S_2. In particular, we get $\sigma_2 \colon E \to \{\top, \bot\}$ with $\sigma_2(x) = \top$ and $\sigma_2(y) = \top$. When the validity of $\phi^{\sigma_1} \vee \phi^{\sigma_2}$ with $\phi^{\sigma_2} := \neg a \wedge b^{\top}$ is checked, we get a counter-example $\rho \colon U^{\sigma_1} \cup U^{\sigma_2} \to \{\top, \bot\}$, from which $\alpha_2 \colon U \to \{\top, \bot\}$, with $\alpha_2(a) = \top$ and $\alpha_2(b) = \top$, can be extracted. Assignment α_2 is added to A_2 leading to a new path in the left expansion tree (Iteration 3 in Figure 2).

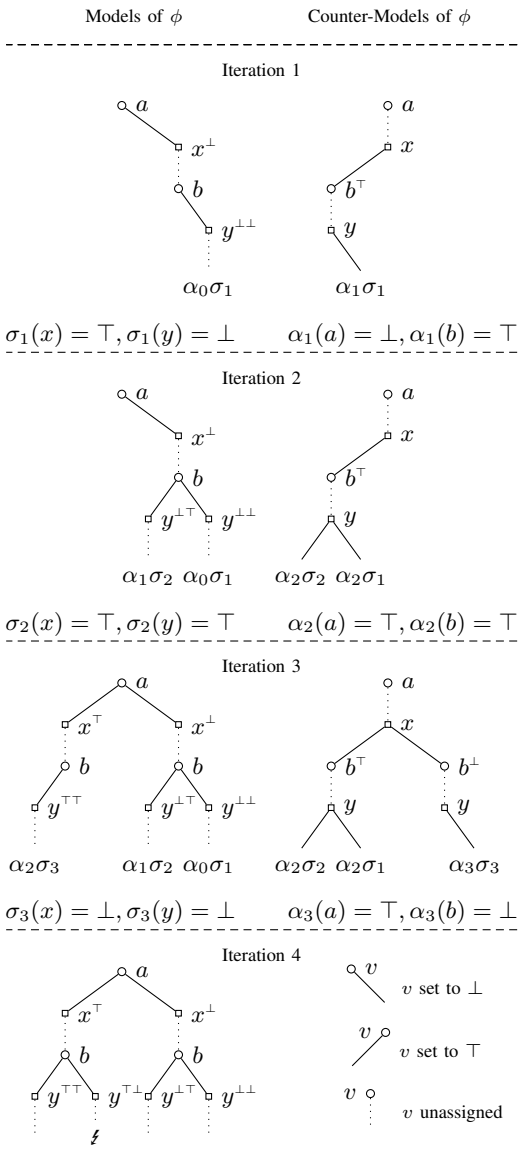

Figure 2: Expansion trees relating the assignments found during solving the QBF $\forall a \exists x \forall b \exists y.\phi$ in Example 5, with initial assignment $\alpha_0(a) = \bot, \alpha_0(b) = \bot$. The assignments shown in the leaves of the trees satisfy (left trees) or falsify (right trees) ϕ.

Iteration 3: Next, $\phi^{\alpha_0} \wedge \phi^{\alpha_1} \wedge \phi^{\alpha_2}$ with $\phi^{\alpha_2} := \neg x^{\top} \wedge \neg y^{\top\top}$ is checked. From model $\tau\colon E^{\alpha_0} \cup E^{\alpha_1} \cup E^{\alpha_2} \to \{\top, \bot\}$, $\sigma_3\colon E \to \{\top, \bot\}$ is extracted, satisfying ϕ^{α_2}. This assignment is different from both σ_1 and σ_2: $\sigma_3(x) = \bot$ and $\sigma_3(y) = \bot$. This again results in a new branch of the expansion tree (see left expansion tree of Iteration 4 in Figure 2). The resulting formula $\phi^{\sigma_1} \vee \phi^{\sigma_2} \vee \phi^{\sigma_3}$ with $\phi^{\sigma_3} := a \wedge b^{\bot}$ is not valid, and from the counter-example $\rho\colon U^{\sigma_1} \cup U^{\sigma_2} \cup U^{\sigma_3} \to \{\top, \bot\}$ we get $\alpha_3\colon U \to \{\top, \bot\}$ with $\alpha_3(a) = \top$ and $\alpha_3(b) = \bot$.

Iteration 4: Finally, the full expansion $\phi^{\alpha_0} \wedge \phi^{\alpha_1} \wedge \phi^{\alpha_2} \wedge \phi^{\alpha_3}$ with $\phi^{\alpha_3} := \bot$ is not satisfiable, meaning that the original formula $\forall a \exists x \forall b \exists y.\phi$ is false.

In the example above we saw that new assignments are generated in each iteration because A_i and S_i build models and counter-models of ϕ. The following definition formalises the relationship between A_i and S_i.

Definition 2. *Let $\Pi.\phi$ be a QBF over universally quantified variables U and existentially quantified variables E. Further, let $A \subseteq \{\alpha \mid \alpha\colon U \mapsto \{\top, \bot\}\}$ and $S \subseteq \{\sigma \mid \sigma\colon E \mapsto \{\top, \bot\}\}$. If for every assignment $\sigma \in S$, there exists an assignment $\alpha \in A$ such that $\alpha\sigma(\neg\phi)$ is true, then we say that A* completes *S. If for every assignment $\alpha \in A$, there exists an assignment $\sigma \in S$ such that $\alpha\sigma(\phi)$ is true, then we say that S* completes *A.*

We now show that S_i completes A_{i-1} and A_i completes S_i if the algorithm does not terminate in iteration i because of the unsatisfiability of the respective expansion.

Lemma 5. *Let $\Pi.\phi$ be a QBF over universally quantified variables U and existentially quantified variables E. Further, let A_{i-1} and A_i with $A_{i-1} \subseteq A_i$ be two sets of full assignments to the universal variables and let S_i be a set of full assignments to the existential variables obtained by iteration i during an execution of the algorithm shown in Figure 1.*

(1) If $\bigwedge_{\alpha \in A_{i-1}} \phi^{\alpha}$ is satisfiable, then S_i completes A_{i-1}, i.e., for every $\mu \in A_{i-1}$, there is an assignment $\nu \in S_i$ such that $\mu\nu(\phi)$ is true.

(2) If $\bigwedge_{\sigma \in S_i} \neg\phi^{\sigma}$ is satisfiable, then A_i completes S_i, i.e., for every $\nu \in S_i$, there is an assignment $\mu \in A_i$ such that $\nu\mu(\neg\phi)$ is true.

Proof. By contradiction. For (1), assume there is an assignment $\mu \in A_{i-1}$ such that there is no assignment $\nu \in S_i$ with $\mu\nu(\phi)$ is true. By assumption $\bigwedge_{\alpha \in A_{i-1}} \phi^{\alpha}$ is satisfiable, so there is a satisfying assignment τ with $\tau|_{E^{\mu}}(\phi^{\mu})$ is true. Then also $\mu(\tau|_{E^{\mu}})^{-\mu}(\phi)$ is true. But $(\tau|_{E^{\mu}})^{-\mu} \in S_i$. For (2), assume that there is an assignment $\mu \in S_i$ such that there is no $\nu \in A_i$ with $\mu\nu(\neg\phi)$ is true. The rest of the argument is similar as in (1). □

Next, we show that the addition of new assignments A' to a set A of universal assignments forces a set S of existential assignments to increase if some completion criteria hold.

Lemma 6. *Let $\Phi = \Pi.\phi$ be a QBF over universally quantified variables U and existentially quantified variables E. Further, let $A \cup A'$ be a set of universal assignments such that $A \cap A' = \emptyset$ and $A' \neq \emptyset$. Let S be a set of existential assignments and assume that $\bigwedge_{\sigma \in S} \neg\phi^{\sigma}$ has the satisfying assignment ρ, $A' \subseteq \{(\rho|_{U^{\sigma}})^{-\sigma} \mid \sigma \in S\}$.*

If S completes A, and $A \cup A'$ completes S, and $\bigwedge_{\alpha \in A \cup A'} \phi^{\alpha}$ evaluates to true under assignment τ, then there exists an assignment $\nu \in \{(\tau|_{E^{\alpha}})^{-\alpha} \mid \alpha \in A \cup A'\}$ with $\nu \notin S$.

Proof. By induction over the number of variables in Π.

Base Case. Assume that Φ has only one variable, i.e., $\Pi = Qx$. Note that $|A'| = 1$ because x is outermost in the prefix and A' is obtained from sub-assignments of ρ. If $Q = \forall$, then the elements of A are full assignments of ϕ, and S is either empty, or it contains the empty assignment $\omega\colon \emptyset \mapsto \{\top, \bot\}$. Let $A' = \{\mu\}$. If S is empty, so is A (because S has to complete A). If τ is a satisfying assignment of ϕ^{μ}, then $\nu = \tau = \omega$ is the empty assignment and $\nu \notin S$. Otherwise, $\omega \in S$. If there is an assignment $\alpha \in A$, then $\phi^{\alpha} \wedge \phi^{\mu}$ is a full expansion of Φ. If this full expansion is true, then $\neg\phi$ is unsatisfiable. Otherwise, $\phi^{\alpha} \wedge \phi^{\mu}$ is unsatisfiable. In both cases, the necessary preconditions for the lemma are not fulfilled. If $A = \emptyset$, then $\mu\omega(\neg\phi)$ is true. Then ϕ^{μ} is unsatisfiable, again violating a precondition. If $Q = \exists$, then $\mu = \omega$ and $A = \emptyset$. If $S = \emptyset$ and $\phi^{\omega} = \phi$ has the satisfying assignment τ, then $\nu = \tau$ and $\nu \notin S$. Otherwise, if there is an assignment $\sigma \in S$, then $\omega\sigma(\neg\phi)$ is true, because $A \cup \{\mu\} = \{\omega\}$ completes S. Hence, if assignment τ satisfies ϕ^{μ}, then $\nu = \tau$, so $\nu \notin S$.

Induction Step. Assume the lemma holds for QBFs with n variables. We show that it also holds for QBFs with $n + 1$ variables. Let $\Phi = Qx\Pi.\phi$ be a QBF over existential variables E and universal variables U with $\Pi = Q_1 x_1 \ldots Q_n x_n$ and $A \cup A'$ and S be as required (S completes A, $A \cup A'$ completes S, $\bigwedge_{\alpha \in A \cup A'} \phi^{\alpha}$ has a satisfying assignment τ, and $\bigwedge_{\sigma \in S} \neg\phi^{\sigma}$ has a satisfying assignment ρ from which A' is obtained).

If $Q = \forall$, then all assignments $\alpha \in A'$ assign the same value t to x, i.e., $\alpha(x) = t$, because these assignments are extracted from assignment ρ and since x is the outermost variable of the prefix of Φ, $\rho(x) = t$. Further, let $A^t = \{\alpha \in A \mid \alpha(x) = t\}$. It is easy to argue that for $\Pi.\phi[x \leftarrow t]$ together with the assignment sets $A^t \cup A'$ and S the induction hypothesis applies, i.e., there is an assignment $\nu \notin S$ with $\nu \in \{(\tau'|_{E^{\alpha}})^{-\alpha} \mid \alpha \in A^t \cup A'\}$ where τ' is the part of τ that satisfies $\bigwedge_{\alpha \in A^t \cup A'} (\phi[x \leftarrow t])^{\alpha}$. Obviously, $\nu \in \{(\tau|_{E^{\alpha}})^{-\alpha} \mid \alpha \in A \cup A'\}$.

If $Q = \exists$, assume that $\tau(x) = t$. Let $\{\sigma \in S \mid \sigma(x) = t\} \subseteq S^t \subseteq S$, and let $A^t \subseteq A$ such that the induction hypothesis applies to $\Pi.\phi[x \leftarrow t]$, $A^t \cup A'$, and S^t. Let τ^t be those sub-assignments of τ that satisfy $\bigwedge_{\alpha \in A^t} \phi^{\alpha}$. Then there is an assignment ν that can be extracted from τ^t with $\nu \notin S^t$. Since $\nu(x) = t$, $\nu \notin S$. This concludes the proof. □

This property also holds in the other direction, i.e., adding a set S' of new assignments to S will force the set A to increase.

Lemma 7. *Let $\Phi = \Pi.\phi$ be a QBF over universally quantified variables U and existentially quantified variables E. Further, let $S \cup S'$ be a set of existential assignments such that $S \cap S' = \emptyset$, $S' \neq \emptyset$, let A be a set of universal assignments, $\bigwedge_{\alpha \in A} \phi^{\alpha}$ has the satisfying assignment τ, $S' \subseteq \{(\tau|_{E^{\alpha}})^{-\alpha} \mid \alpha \in A\}$.*

If A completes S and $S \cup S'$ completes A and $\bigwedge_{\sigma \in S \cup S'} \neg\phi^{\sigma}$ evaluates to true under assignment ρ, then there exists an

assignment $\nu \in \{(\rho|_{U^\sigma})^{-\sigma} \mid \sigma \in S \cup S'\}$ *with* $\nu \notin A$.

Proof. The proof is analogous to the proof of Lemma 6. □

Now that we have identified the relations between the sets of universal and existential assignments, we use them to show that the algorithm from Figure 1 terminates.

Theorem 2. *The algorithm shown in Figure 1 terminates for any QBF* $\Phi = \Pi.\phi$.

Proof. By induction over the number of iterations i, we argue that sets $A_{i-1} \subset A_i$ and $S_{i-1} \subset S_i$.

Base Case. Let $i = 1$ and $A_0 = \{\alpha_0\}$. $S_0 \subset S_1$, because $S_0 = \emptyset$ and $\sigma_1 \in S_1$ is a satisfying assignment of ϕ^{α_0} (if ϕ^{α_0} is unsatisfiable, the algorithm terminates). $A_0 \subset A_1$ directly follows from Lemma 4.

Induction Step. For $i > 1$, we argue that $S_i \subset S_{i+1}$. By induction hypothesis the theorem holds for iteration i, i.e., $A_i = A_{i-1} \cup A'$ with $A_{i-1} \cap A' = \emptyset$ and $A' \neq \emptyset$ and $S_i = S_{i-1} \cup S'$ with $S_{i-1} \cap S' = \emptyset$ and $S' \neq \emptyset$. Because of Lemma 5, S_i completes A_{i-1}, and A_i completes S_i. Furthermore, if $\bigwedge_{\sigma \in S_i} \neg\phi^\sigma$ is satisfiable under some assignment ρ (otherwise the algorithm would terminate), by construction $A' \subseteq \{(\rho|_{U^\sigma})^{-\sigma} \mid \sigma \in S_i\}$. Hence, Lemma 6 applies and if $\bigwedge_{\alpha \in A_i} \phi^\alpha$ is satisfiable under some assignment τ (otherwise the algorithm would immediately terminate), then there is an assignment $\nu \in \{(\tau|_{E^\alpha})^{-\alpha} \mid \alpha \in A_i\}$ with $\nu \notin S_i$.

The argument for $A_i \subset A_{i+1}$ is similar and uses the property shown in Lemma 7. □

Note that the algorithm presented above does not make any assumptions on the formula structure, i.e., for a QBF $\Pi.\phi$ it is not required that ϕ is in conjunctive normal form. Without any modification, our algorithm also works on formulas in PCNF—as SAT solvers typically process formulas in CNF only, we focus on this representation for the rest of the paper.

We conclude this section by arguing that the ∀Exp+Res [5], [28], [29] calculus yields the theoretical foundation of our algorithm for refuting a formula $\Pi.\phi$ in PCNF with universal variables U. The ∀Exp+Res calculus consists of two rules, the axiom rule

$$\frac{}{C^\alpha}$$

where C is a clause of ϕ and $\alpha\colon U \to \{\top, \bot\}$ is a universal assignment as well as the resolution rule

$$\frac{C_1 \vee x^\omega \qquad C_2 \vee \neg x^\omega}{C_1 \vee C_2}$$

A derivation in ∀Exp+Res is a sequence of clauses where each clause is either obtained by the axiom or derived from previous clauses by the application of the resolution rule. A refutation of a PCNF $\Pi.\phi$ is a derivation of the empty clause.

The application of the axiom instantiates the universal variables of one clause of ϕ. If enough of these instantiations can be found in order to derive the empty clause by the application of the resolution rule, the QBF $\Pi.\phi$ is false. Our algorithm in Figure 1 does not instantiate individual clauses, but all clauses of ϕ at once with a particular assignment of the universal variables. Hence, when the SAT solver finds $\psi_\forall = \bigwedge_{\alpha \in A_i} \phi^\alpha$ unsatisfiable for some A_i, not necessarily all clauses of $\psi_\forall$ are required to derive the empty clause via resolution, but only the minimal unsatisfiable core of $\psi_\forall$, i.e., a subset of the clauses such that the removal of any clause would make this formula satisfiable.

Proposition 1. *Let* $\Pi.\phi$ *be a false QBF. Further, let* $\psi_\forall = \bigwedge_{\alpha \in A_i} \phi^\alpha$ *be obtained by the application of the algorithm in Figure 1. Further, let* $\psi'_\forall$ *be an unsatisfiable core of* $\psi_\forall$. *Then there is a* ∀*Exp+Res refutation such that all clauses that are introduced by the axiom rule occur in* $\psi'_\forall$.

VI. Implementation

The algorithm described in Section V is realised in the solver ijtihad[2] The most recent version of ijtihad is available at

https://extgit.iaik.tugraz.at/scos/ijtihad

The solver is implemented in C++ and currently processes formulas in PCNF available in the QDIMACS format. For accessing SAT solvers, ijtihad uses the IPASIR interface [4], which makes changing the SAT solver very easy. The SAT solver used in all of our experiments is Glucose [1]. Although the base implementation does reasonably well, we have realised various optimizations to make ijtihad even more viable in practice. Some of them are discussed in the following.

For solving a QBF $\Pi.\phi$, the basic algorithm shown in Figure 1 adds instantiations of ϕ to $\psi_\forall = \bigwedge_{\alpha \in A_{i-1}} \phi^\alpha$ and $\psi_\exists = \bigwedge_{\sigma \in S_i} \neg\phi^\sigma$ in each iteration i until the formula is decided. The calls to the SAT solver in Line 5 and Line 8 are done incrementally, i.e., we create two instances of the SAT solver and provide them with the clauses stemming from new instantiations of ϕ at each iteration. For simplicity, we omit indices of sets A and S and refer to an arbitrary iteration of the execution of the algorithm in the following discussion.

Figure 5 relates set sizes of A and S as well as the accumulated time that one SAT solver needs to solve $\psi_\forall$ with the time the other SAT solver needs to solve $\psi_\exists$ for the formulas of the PCNF track of QBFEVAL'17 (preprocessed with Bloqqer [6]). We also distinguish between true and false formulas. In Figure 3 we see that for true formulas, set S tends to be larger than A, while for false instances the picture is less clear. Figure 4 shows the overall time needed for solving $\psi_\forall$ (y-axis) and $\psi_\exists$ (x-axis). In almost all cases, the solver that handles $\psi_\forall$ needs more time than the solver that handles $\psi_\exists$. This may be founded on the observation that many QBFs have considerably more existential variables than universal variables [37], hence the instantiations added to $\psi_\forall$ are much larger than the instantiations added to $\psi_\exists$.

In Line 1 of Figure 1, the set of universal assignments A is initialised with *one* arbitrary assignment α_0. Obviously, the set A may also be initialized with multiple assignments. In our current implementation, we initialize A with the assignments that set the variables of one universal quantifier block to $\bot$ and the variables of all other universal quantifier blocks to $\top$.

[2]The name ijtihad refers to the effort of solving cases in Islamic law (for details see https://en.wikipedia.org/wiki/Ijtihad).

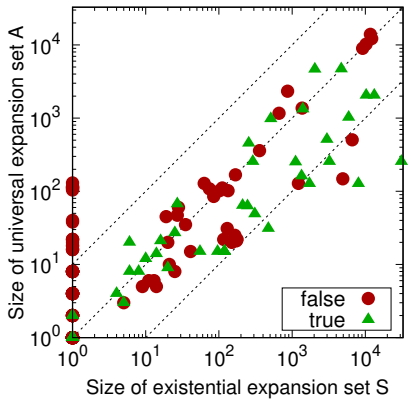

Figure 3: Sizes of sets S and A

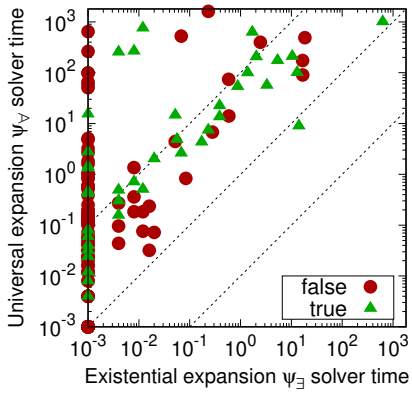

Figure 4: Time used for solving $\psi_\exists$ and $\psi_\forall$

Figure 5: Set sizes and time consumed during SAT calls for solved instances from QBFEVAL'17 preprocessed by Bloqqer

The impact of various initialization heuristics remains to be investigated in future work.

In Line 7 and Line 10 our algorithm increases the size of S and A in each iteration of the main loop, as argued in Theorem 2. In the worst case, this leads to an exponential increase in space consumption. Although we detect shared clauses among the instantiations, that alone is not enough to significantly reduce the space consumption. However, some of the assignments found in an earlier iteration could become obsolete after better assignments were found. It is therefore beneficial to empty either S or A and then reconstruct them from $\psi_\forall$ and $\psi_\exists$, similarly to what is done in Line 7 and Line 10. We evaluated several heuristics for scheduling these set resets, and we found that resetting periodically and close to the memory limit works best. The regular resetting of one set has a similar effect as restarts in SAT solvers, and we observed a considerable improvement in performance, especially in terms of memory consumption. Our implementation periodically resets the set A, since experiments indicate that the resulting formula $\psi_\forall$ is much harder to solve than $\psi_\exists$ as seen in Figure 4. Besides the aforementioned imbalance between universal and existential variables, it is also likely due to the structure of $\psi_\exists$ which is a conjunction of formulas in disjunctive normal form. Note that this reset of A does not affect the termination argument presented in Theorem 2, since the sets A and S still complete each other.

Finally, we extended the presented approach with orthogonal reasoning techniques like QCDCL [21] for exploiting the different strengths of ∀Exp+Res and Q-resolution, yielding a hybrid solver that smoothly integrates both solving paradigms. To this end, we implemented the prototypical solver called Heretic which pursues the following idea: The main loop of the algorithm shown in Figure 1 (Lines 4-12) is extended in a sequential portfolio style such that a QCDCL solver is periodically called. After each call, all clauses that were learned through QCDCL are added to $\Pi.\Phi$, making them available in further iterations. These new clauses potentially exclude assignments that would otherwise be possible and that could result in more iterations of the main loop.

The solver Heretic extends Ijtihad by additional invocations of the QCDCL solver DepQBF [36]. About every 30 seconds, DepQBF is called and run for about 30 seconds. The learnt clauses are obtained via the API of DepQBF. Leveraging learned cubes is subject to future work.

VII. Evaluation

We evaluate non-recursive expansion as implemented in our solvers Ijtihad and its hybrid variant Heretic on the benchmarks from the PCNF track of the QBFEVAL'17 competition. All experiments were carried out on a cluster of Intel Xeon CPUs (E5-2650v4, 2.20 GHz) running Ubuntu 16.04.1 with a CPU time limit of 1800 seconds and a memory limit of 7 GB. We considered the following top-performing solvers from QBFEVAL'17: Qute [38], Rev-Qfun [25], RAReQS [26], CAQE [39], [43], DynQBF [12], GhostQ [26], [34], DepQBF [36], QESTO [30], and QSTS [9], [10]. Our experiments are based on original benchmarks without preprocessing and benchmarks preprocessed using Bloqqer [6], [23] with a timeout of two hours.[3] We included the 76 formulas already solved by Bloqqer in both benchmark sets.

Tables I and II show the total numbers of solved instances (S), solved unsatisfiable ($\bot$) and satisfiable ones ($\top$), and total CPU time including timeouts. In the following, we focus on a comparison of our solvers Ijtihad and Heretic with RAReQS (cf. Figure 6). Unlike our solvers, RAReQS is based on a recursive implementation of expansion.

In general, preprocessing has a considerable impact on the number of solved instances. The difference in solved instances between Ijtihad and RAReQS is 17 on original instances (Table I), and becomes larger on preprocessed instances (Table II).

Notably Heretic, despite its simple design, significantly outperforms Ijtihad on the two benchmark sets. Moreover, Heretic is ranked third on preprocessed instances (Table II) and thus is on par with state-of-the-art solvers. On the two benchmark sets, the gap in solved instances between RAReQS and Heretic is considerably smaller than the one between RAReQS and Ijtihad.

We report on memory consumption of expansion-based solvers. While RAReQS, Ijtihad, and Heretic run out of

[3] We refer to an online appendix [7] for additional experiments.

Solver	*S*	⊥	⊤	*Time*
Rev-Qfun	220	145	75	572K
GhostQ	194	120	74	617K
CAQE	170	128	42	656K
RAReQS	167	133	34	660K
DepQBF	167	121	46	666K
Heretic	163	133	30	664K
QSTS	152	116	36	687K
Ijtihad	150	127	23	684K
Qute	130	91	39	720K
QESTO	109	86	23	761K
DynQBF	72	38	34	826K

TABLE I: Original instances.

Solver	*S*	⊥	⊤	*Time*
RAReQS	256	180	76	508K
CAQE	251	168	83	522K
Heretic	245	172	73	522K
Rev-Qfun	219	148	71	568K
Ijtihad	217	156	61	564K
QSTS	208	151	57	585K
QESTO	196	137	59	610K
DepQBF	183	117	66	633K
GhostQ	163	100	63	670K
Qute	154	109	45	682K
DynQBF	151	95	56	684K

TABLE II: Preprocessing by Bloqqer.

Solver	*S*	⊥	⊤	*Time*
Heretic	92	81	11	195K
DepQBF	89	74	15	205K
CAQE	88	73	15	204K
RAReQS	82	78	4	211K
QSTS	81	69	12	216K
Ijtihad	80	73	7	212K
Rev-Qfun	78	75	3	224K
Qute	70	60	10	236K
QESTO	51	44	7	269K
GhostQ	45	39	6	279K
DynQBF	15	13	2	332K

TABLE III: 197 original instances with four or more quantifier blocks.

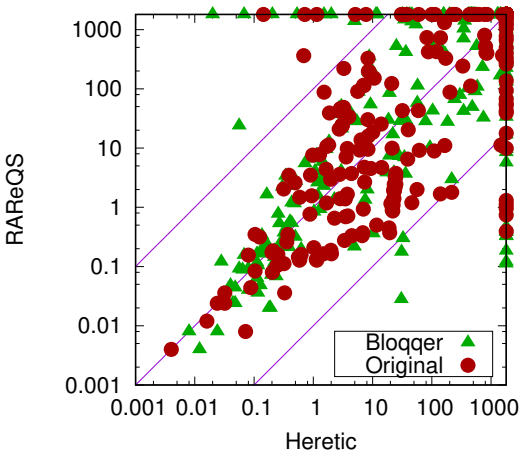

Figure 6: Scatter plots of the run times of Heretic and RAReQS on original instances (related to Table I) and on instances preprocessed by Bloqqer (related to Table II).

memory on 42, 61, and 39 original instances (Table I), respectively, these numbers drop to 17, 41, and 24, respectively, with preprocessing (Table II). The average memory footprint is 1718 MB, 1836 MB, and 1842 MB for RAReQS, Ijtihad, and Heretic, respectively, and 1056 MB, 1311 MB, and 1187 MB on preprocessed instances. Interestingly, Ijtihad has a smaller median memory footprint than RAReQS without (792 MB vs. 802 MB) and with preprocessing (286 MB vs. 364 MB).

The strength of Heretic becomes obvious for formulas that have four or more quantifier blocks (i.e., three or more quantifier alternations), cf. [37]. As shown in Table III, Heretic outperforms all other solvers on these instances. We made a similar observation on preprocessed formulas.

Moreover, Heretic solves only four original instances less than DepQBF (Table I), and outperforms DepQBF on preprocessed instances (Table II). These results indicate the potential of combining the orthogonal proof systems ∀Exp+Res as implemented in Ijtihad and Q-resolution as implemented in DepQBF in a hybrid solver such as Heretic.

Although RAReQS outperforms both Ijtihad and Heretic on the two given benchmark sets (Tables I and II), RAReQS failed to solve certain instances that were solved by Ijtihad and Heretic. Table IV shows related statistics. E.g., on preprocessed instances (row "B"), 218 instances were solved

	R	*vs.*	*I*	*R*	*vs.*	*H*	*I*	*vs.*	*H*	*D*	*vs.*	*H*
	<	=	>	<	=	>	<	=	>	<	=	>
N	27	140	10	26	141	22	5	145	18	65	102	61
B	56	200	17	38	218	27	7	210	35	17	166	79

TABLE IV: Pairwise comparison of RAReQS (R), Ijtihad (I), Heretic (H), and DepQBF (D) by instances without (N) and with preprocessing by Bloqqer (B) that were solved by only one solver of the considered pair (<, >) or by both (=).

by both RAReQS and Heretic (column "*R vs. H*), 38 only by RAReQS, and 27 only by Heretic. Summing up these numbers yields a total of 283 solved instances (more than any individual solver on preprocessed instances in Table II) that could have been solved by a *hypothetical solver* combining RAReQS and Heretic. This observation underlines the strength of expansion in general and, in particular, of the hybrid approach implemented in Heretic. Heretic solved a significant amount of instances not solved by RAReQS, it clearly outperformed Ijtihad on all benchmarks (column "*I vs. H*") and DepQBF on preprocessed ones ("*D vs. H*").

VIII. Conclusion

We presented a novel non-recursive algorithm for expansion-based QBF solving that uses only two SAT solvers for incrementally refining the propositional abstraction and the negated propositional abstraction of a QBF. We gave a concise proof of termination and soundness and demonstrated with several experiments that our prototype compares well with the state of the art. In addition to non-recursive expansion, we also studied the impact of combining Q-resolution and ∀Exp+Res in a hybrid approach. To this end, we coupled a QCDCL solver and non-recursive expansion to make clauses derived by the QCDCL solver available to the expansion solver. Experimental results indicated that the hybrid approach significantly outperforms our implementation of non-recursive expansion indicating the potential of combining expansion-based approaches with Q-resolution which gives rise to an exciting direction of future work. Further, our current implementation supports only formulas in conjunctive normal form while in theory, our approach does not make any assumptions on the structure of the propositional part of the QBF. We also plan to investigate how this formula structure can be exploited for efficiently processing the negation of the formula.

References

[1] Audemard, G., Simon, L.: Predicting learnt clauses quality in modern SAT solvers. In: IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11-17, 2009. pp. 399–404 (2009), http://ijcai.org/Proceedings/09/Papers/074.pdf

[2] Ayari, A., Basin, D.A.: QUBOS: Deciding Quantified Boolean Logic Using Propositional Satisfiability Solvers. In: FMCAD. LNCS, vol. 2517, pp. 187–201. Springer (2002)

[3] Balabanov, V., Jiang, J.R., Scholl, C., Mishchenko, A., Brayton, R.K.: 2QBF: Challenges and Solutions. In: SAT. LNCS, vol. 9710, pp. 453–469. Springer (2016)

[4] Balyo, T., Biere, A., Iser, M., Sinz, C.: SAT Race 2015. Artif. Intell. 241, 45–65 (2016), http://dx.doi.org/10.1016/j.artint.2016.08.007

[5] Beyersdorff, O., Chew, L., Janota, M.: On unification of QBF resolution-based calculi. In: Proc. of the 39th Int. Symbosium on Mathematical Foundations of Computer Science (MFCS). LNCS, vol. 8635, pp. 81–93. Springer (2014)

[6] Biere, A., Lonsing, F., Seidl, M.: Blocked Clause Elimination for QBF. In: CADE. LNCS, vol. 6803, pp. 101–115. Springer (2011)

[7] Bloem, R., Braud-Santoni, N., Hadzic, V., Egly, U., Lonsing, F., Seidl, M.: Expansion-Based QBF Solving Without Recursion. CoRR abs/1807.08964 (2018), https://arxiv.org/abs/1807.08964, FMCAD 2018 proceedings version with appendix

[8] Bloem, R., Könighofer, R., Seidl, M.: SAT-Based Synthesis Methods for Safety Specs. In: VMCAI. LNCS, vol. 8318, pp. 1–20. Springer (2014)

[9] Bogaerts, B., Janhunen, T., Tasharrofi, S.: SAT-to-SAT in QBFEval 2016. In: QBF Workshop. CEUR Workshop Proceedings, vol. 1719, pp. 63–70. CEUR-WS.org (2016)

[10] Bogaerts, B., Janhunen, T., Tasharrofi, S.: Solving QBF Instances with Nested SAT Solvers. In: Beyond NP. AAAI Workshops, vol. WS-16-05. AAAI Press (2016)

[11] Bubeck, U., Kleine Büning, H.: Bounded Universal Expansion for Preprocessing QBF. In: SAT. LNCS, vol. 4501, pp. 244–257. Springer (2007)

[12] Charwat, G., Woltran, S.: Dynamic Programming-based QBF Solving. In: QBF Workshop. CEUR Workshop Proceedings, vol. 1719, pp. 27–40. CEUR-WS.org (2016)

[13] Cheng, C., Hamza, Y., Ruess, H.: Structural Synthesis for GXW Specifications. In: CAV. LNCS, vol. 9779, pp. 95–117. Springer (2016)

[14] Cheng, C., Lee, E.A., Ruess, H.: autoCode4: Structural Controller Synthesis. In: TACAS. LNCS, vol. 10205, pp. 398–404. Springer (2017)

[15] Clarke, E.M., Grumberg, O., Jha, S., Lu, Y., Veith, H.: Counterexample-Guided Abstraction Refinement for Symbolic Model Checking. J. ACM 50(5), 752–794 (2003)

[16] Dershowitz, N., Hanna, Z., Katz, J.: Bounded Model Checking with QBF. In: SAT. LNCS, vol. 3569, pp. 408–414. Springer (2005)

[17] Egly, U., Kronegger, M., Lonsing, F., Pfandler, A.: Conformant Planning as a Case Study of Incremental QBF Solving. Ann. Math. Artif. Intell. 80(1), 21–45 (2017)

[18] Faymonville, P., Finkbeiner, B., Rabe, M.N., Tentrup, L.: Encodings of Bounded Synthesis. In: TACAS. LNCS, vol. 10205, pp. 354–370. Springer (2017)

[19] Finkbeiner, B., Tentrup, L.: Detecting Unrealizability of Distributed Fault-tolerant Systems. Logical Methods in Computer Science 11(3) (2015)

[20] Gascón, A., Tiwari, A.: A Synthesized Algorithm for Interactive Consistency. In: NASA Formal Methods. LNCS, vol. 8430, pp. 270–284. Springer (2014)

[21] Giunchiglia, E., Marin, P., Narizzano, M.: Reasoning with Quantified Boolean Formulas. In: Handbook of Satisfiability, FAIA, vol. 185, pp. 761–780. IOS Press (2009)

[22] Giunchiglia, E., Marin, P., Narizzano, M.: sQueezeBF: An Effective Preprocessor for QBFs Based on Equivalence Reasoning. In: SAT. LNCS, vol. 6175, pp. 85–98. Springer (2010)

[23] Heule, M., Järvisalo, M., Lonsing, F., Seidl, M., Biere, A.: Clause Elimination for SAT and QSAT. JAIR 53, 127–168 (2015)

[24] Heyman, T., Smith, D., Mahajan, Y., Leong, L., Abu-Haimed, H.: Dominant Controllability Check Using QBF-Solver and Netlist Optimizer. In: SAT. LNCS, vol. 8561, pp. 227–242. Springer (2014)

[25] Janota, M.: Towards Generalization in QBF Solving via Machine Learning. In: AAAI. AAAI Press (2018)

[26] Janota, M., Klieber, W., Marques-Silva, J., Clarke, E.: Solving QBF with Counterexample Guided Refinement. Artif. Intell. 234, 1–25 (2016)

[27] Janota, M., Klieber, W., Marques-Silva, J., Clarke, E.M.: Solving QBF with Counterexample Guided Refinement. In: SAT. LNCS, vol. 7317, pp. 114–128. Springer (2012)

[28] Janota, M., Marques-Silva, J.: On Propositional QBF Expansions and Q-Resolution. In: SAT. LNCS, vol. 7962, pp. 67–82. Springer (2013)

[29] Janota, M., Marques-Silva, J.: Expansion-Based QBF Solving versus Q-Resolution. Theor. Comput. Sci. 577, 25–42 (2015)

[30] Janota, M., Marques-Silva, J.: Solving QBF by Clause Selection. In: IJCAI. pp. 325–331. AAAI Press (2015)

[31] Janota, M., Silva, J.P.M.: Abstraction-Based Algorithm for 2QBF. In: SAT. LNCS, vol. 6695, pp. 230–244. Springer (2011)

[32] Kleine Büning, H., Bubeck, U.: Theory of Quantified Boolean Formulas. In: Handbook of Satisfiability, FAIA, vol. 185, pp. 735–760. IOS Press (2009)

[33] Kleine Büning, H., Karpinski, M., Flögel, A.: Resolution for Quantified Boolean Formulas. Inf. Comput. 117(1), 12–18 (1995)

[34] Klieber, W., Sapra, S., Gao, S., Clarke, E.M.: A Non-Prenex, Non-Clausal QBF Solver with Game-State Learning. In: SAT. LNCS, vol. 6175, pp. 128–142. Springer (2010)

[35] Letz, R.: Lemma and Model Caching in Decision Procedures for Quantified Boolean Formulas. In: TABLEAUX. LNCS, vol. 2381, pp. 160–175. Springer (2002)

[36] Lonsing, F., Egly, U.: DepQBF 6.0: A Search-Based QBF Solver Beyond Traditional QCDCL. In: CADE. LNCS, vol. 10395, pp. 371–384. Springer (2017)

[37] Lonsing, F., Egly, U.: Evaluating QBF Solvers: Quantifier Alternations Matter. In: CP. LNCS, vol. 11008, pp. 276–294. Springer (2018)

[38] Peitl, T., Slivovsky, F., Szeider, S.: Dependency Learning for QBF. In: SAT. LNCS, vol. 10491, pp. 298–313. Springer (2017)

[39] Rabe, M.N., Tentrup, L.: CAQE: A Certifying QBF Solver. In: FMCAD. pp. 136–143. IEEE (2015)

[40] Ranjan, D.P., Tang, D., Malik, S.: A Comparative Study of 2QBF Algorithms. In: SAT (2004)

[41] Rintanen, J.: Asymptotically Optimal Encodings of Conformant Planning in QBF. In: AAAI. pp. 1045–1050. AAAI Press (2007)

[42] Tentrup, L.: Non-prenex QBF solving using abstraction. In: SAT. LNCS, vol. 9710, pp. 393–401. Springer (2016)

[43] Tentrup, L.: On Expansion and Resolution in CEGAR Based QBF Solving. In: CAV. LNCS, vol. 10427, pp. 475–494. Springer (2017)

[44] Tu, K., Hsu, T., Jiang, J.R.: QELL: QBF Reasoning with Extended Clause Learning and Levelized SAT Solving. In: SAT. LNCS, vol. 9340, pp. 343–359. Springer (2015)

[45] Vizel, Y., Weissenbacher, G., Malik, S.: Boolean Satisfiability Solvers and Their Applications in Model Checking. Proceedings of the IEEE 103(11), 2021–2035 (2015)

[46] Wimmer, R., Reimer, S., Marin, P., Becker, B.: HQSpre - An Effective Preprocessor for QBF and DQBF. In: TACAS. LNCS, vol. 10205, pp. 373–390. Springer (2017)

[47] Zhang, L., Malik, S.: Towards a Symmetric Treatment of Satisfaction and Conflicts in Quantified Boolean Formula Evaluation. In: CP. LNCS, vol. 2470, pp. 200–215. Springer (2002)

[48] Zhang, W.: QBF Encoding of Temporal Properties and QBF-Based Verification. In: IJCAR. LNCS, vol. 8562, pp. 224–239. Springer (2014)

APPENDIX A
ADDITIONAL EXPERIMENTAL DATA

Preprocessing with HQSpre

Solver	*S*	⊥	⊤	*Time*
CAQE	306	197	109	415K
RAReQS	294	194	100	429K
QESTO	287	194	93	443K
Rev-Qfun	281	190	91	453K
Heretic	279	188	91	460K
QSTS	264	179	85	484K
DepQBF	263	176	87	490K
Qute	255	171	84	497K
Ijtihad	250	176	74	500K
GhostQ	244	163	81	522K
DynQBF	233	156	77	528K

TABLE V: Preprocessing by HQSpre.

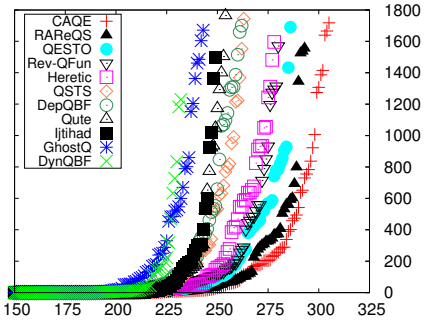

Figure 7: Plot related to Table V.

Solver	*S*	⊥	⊤	*Time*
DepQBF	33	23	10	123K
Heretic	30	20	10	127K
QESTO	29	18	11	127K
CAQE	29	16	13	127K
Qute	24	17	7	137K
RAReQS	23	17	6	136K
Rev-Qfun	22	18	4	137K
QSTS	20	12	8	140K
Ijtihad	17	11	6	145K
GhostQ	11	4	7	160K
DynQBF	7	4	3	163K

TABLE VI: Preprocessing by HQSpre: 97 instances with four or more quantifier blocks.

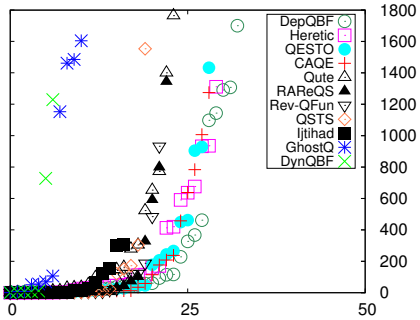

Figure 8: Plot related to Table VI.

	R	*vs.*	*I*	*R*	*vs.*	*H*	*I*	*vs.*	*H*
	<	=	>	<	=	>	<	=	>
No preprocessing	27	140	10	26	141	22	5	145	18
Bloqqer	56	200	17	38	218	27	7	210	35
HQSpre	54	240	10	40	254	25	1	249	30

TABLE VII: Pairwise comparison of RAReQS (R), Ijtihad (I), and Heretic (H) by numbers of instances from QBFEVAL'17 with and without preprocessing that were solved by only one solver of the considered pair (<, >) or by both (=).

Preprocessing with Bloqqer

Solver	*S*	⊥	⊤	*Time*
Heretic	86	71	15	130K
RAReQS	79	72	7	143K
CAQE	73	59	14	161K
QSTS	70	59	11	160K
Ijtihad	67	57	10	158K
Rev-Qfun	64	55	9	168K
DepQBF	61	44	17	178K
QESTO	51	43	8	190K
Qute	46	40	6	208K
GhostQ	26	16	10	231K
DynQBF	23	13	10	241K

TABLE VIII: 154 preprocessed instances with four or more quantifier blocks.

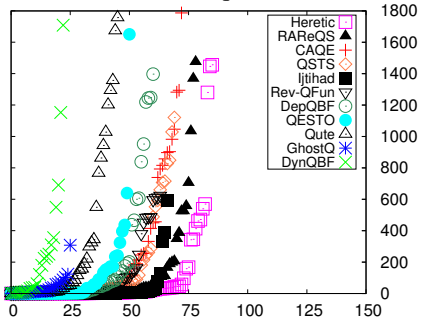

Figure 9: Plot related to Table VIII.

Additional plots

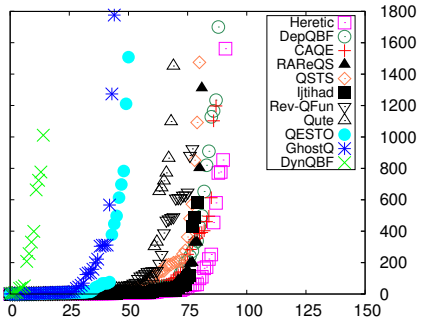

Figure 10: Plot related to Table III.

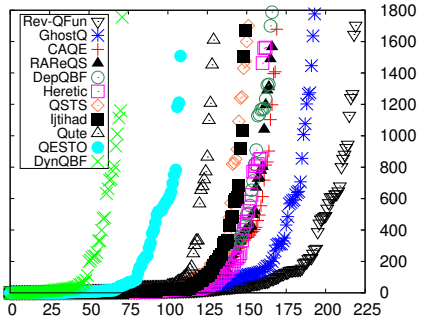

Figure 11: Plot related to Table I.

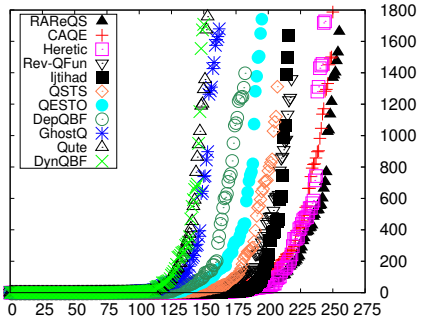

Figure 12: Plot related to Table II.