OuterCount: A First-Level Solution-Counter for Quantified Boolean Formulas*

Ankit Shukla[1][0000−0002−1038−3602], Sibylle Möhle[1][0000−0001−7883−7749], Manuel Kauers[2][0000−0001−8641−6661], and Martina Seidl[3][0000−0002−3267−4494]

[1] Institute for Formal Models and Verification, JKU Linz, Austria
[2] Institute for Algebra, JKU Linz, Austria,
[3] Institute for Symbolic Artificial Intelligence, JKU Linz, Austria
{ankit.shukla, sibylle.moehle-rotondi, manuel.kauers, martina.seidl}@jku.at

Abstract. Counting the solutions of symbolic encodings is an intriguing computational problem with many applications. In the field of propositional satisfiability (SAT) solving, for example, many algorithms and tools have emerged to tackle the counting problem. For quantified Boolean formulas (QBFs), an extension of SAT with quantifiers used to compactly encode and solve problems of formal verification, synthesis, planning, etc., practical solution counting has not been considered yet.
We present the first practical counting algorithm for top-level solutions. We prove soundness of our algorithm for true and false formulas and show how to implement it with recent QBF solving technology. Our evaluation of benchmarks from the recent QBF competition gives promising results for this difficult problem.

Keywords: Quantified Boolean Formulas · Solution Counting · #QBF.

1 Introduction

QBF solution counting [21], also known as #QBF, is the problem of computing the number of *(counter-)models* of a given *quantified Boolean formula* (QBF). Like #SAT [16], the counting problem for propositional satisfiability (SAT), #QBF is considered to be very hard. It belongs to #PSPACE [21], an intractable problem class. As #SAT is an essential task in many application domains, including probabilistic reasoning [10, 33], the analysis of software vulnerability [8, 38], and the verification of neural networks [3, 29], numerous approaches exist to practically solve #SAT and variations of this problem [11, 16]. In contrast, the #QBF problem has only been studied theoretically [17]. Practical counting tools do not exist so far although similar applications as for #SAT can be expected. With this work, we start to close this gap by presenting the first QBF counter for a large class of QBF problems.

The presence of quantifiers over the Boolean variables pose theoretical as well as engineering challenges to uplifting techniques straightway from propositional

* This work has been supported by the Austrian Science Fund (FWF) under projects W1255-N23 and P31571-N32, the LIT AI Lab funded by the State of Upper Austria.

logic. The universal and existential quantifiers render the decision problem of QBF PSPACE-complete [35], making QBFs a natural choice for encoding various problems from verification and artificial intelligence [34, 32]. Because of the quantification, we observe a duality between true and false problems. This duality does not exist in SAT. If a propositional formula is satisfiable, there exists at least one model, i.e., a truth assignment to the propositional variables under which the formula evaluates to true. If a propositional formula is unsatisfiable, no satisfying assignment exists, i.e., all assignments are counter-models. In contrast, models of a true QBF are functions that tell us how to set the existential variables based on given values of the universal variables. Dually, counter-models of a false QBF are functions that tell us how to set the universal variables based on given values of the existential variables. In both cases, these functions reflect the quantifier dependencies. As existential (resp. universal) variables in the outermost quantifier block do not have any dependencies, their solutions are simply Boolean values in the case of true (resp. false) formulas. Concrete examples of interesting QBF encodings are bounded model checking, planning or formal synthesis. For many of these encodings, the solutions to these problems (e.g., error traces, plans, synthesized programs) are Boolean assignments of the variables in the outermost quantifier block. We are concerned with the question how many such solutions exist for a given QBF.

In this work, we present a concise formulation of the solution counting problem for the variables of the outermost quantifier block and solve it with an enumeration-based approach. The assignments of these variables are of particular interest because their values indicate the solutions to many application problems, e.g., the synthesized implementation in reactive synthesis, the error traces in bounded model checking, or the plans in planning. We implemented our method with recent QBF solving technology and further evaluated our implementation on recent QBF competition benchmarks.

The organization of the paper is as follows. After discussing related work and preliminaries in Section 2 and in Section 3, we formalize our model-counting approach for true formulas in Section 4. On this basis, we present the dual approach for false formulas in Section 5. This approach involves some additional transformation steps not necessary for true formulas. Both are implemented with recent QBF solving technology. In Section 7, we evaluate our tool on the benchmarks from the last QBF competition. In Section 8, we discuss possible future work.

2 Related Work

For many exact counting problems, enumerative approaches are among the first techniques in order to realize counting tools. This applies, for example, to model counting for SAT [9, 13], to the counting of minimal inconsistent sets [19, 25], to the counting of unsatisfiable sets [2, 6, 36] and satisfiable sets [7], to counting in answer set programming [14], and to the counting of minimal correction sets [27, 28].

An enumeration-based model counter iteratively adds the negation of already found models in the form of so-called *blocking clauses* to the given formula. The addition of blocking clauses excludes the respective models from the solution space until no further models can be found. In the case that a problem has a huge amount of solutions, often such approaches are run only until a certain time limit is reached. In the context of QBF solving, the idea of adding blocking clauses is also known as good (or solution) learning [15, 22, 37], but it has not been used for counting. For SAT, Bayardo and Pehoushek [4] used good learning and the minimization of goods in the context of model counting.

An enumeration-based approach to #SAT was used to enable the formal proof of its correctness [26]. In principle, the proof consists in showing that the detected models are pairwise contradicting and that all models are found. From this, it follows that the corresponding model count is correct. Modern counting tools implement alternative techniques like the detection of connected components, tight integration into the solving process, or hashing solutions (see [11, 16] for more details on propositional model counting).

The enumeration of all models is a very closely related problem. For propositional logic, this problem is known as ALLSAT. In the context of QBF, Ehlers et al. [5] defined the ALLQBF problem as the task to find all assignments of free variables occurring in a given QBF such that the formula evaluates to true. In their work, they realize different learning approaches, including one which iteratively builds a representation of all satisfying assignments in disjunctive normal form (DNF). While they are not interested in calculating the model count, they aim for a DNF representation that is as small as possible while describing all models. Also, they do not consider false formulas.

3 Preliminaries

We consider QBFs of the form $\Pi.\phi$ that consist of a *(quantifier) prefix* $\Pi = Q_1X_1 \dots Q_nX_n$ (where $Q_i \in \{\forall, \exists\}$, $Q_i \neq Q_{i+1}$, and $X_1, \dots, X_n$ are pairwise disjoint and non-empty sets of variables) and the *matrix* ϕ, a propositional formula over variables X_i. We call i the *(quantifier) level* of *quantifier block* Q_iX_i, e.g., variables of X_1 are at level 1 of prefix $Q_1X_1 \dots Q_nX_n$. If $x \in X_i$ and $y \in X_j$ and $i < j$, then $x_i < x_j$. For the propositional part of a QBF, we use the standard Boolean connectives $\wedge$ (conjunction), $\vee$ (disjunction), $\rightarrow$ (implication), $\leftrightarrow$ (equivalence), and $\neg$ (negation). A QBF $\Pi.\phi$ is in *prenex conjunctive normal form* (PCNF) if ϕ is a conjunction of clauses. A *clause* is a disjunction of literals and a literal is a variable or a negated variable. For a literal l, $var(l) = x$ if $l = x$ or $l = \neg x$. A *unit clause* contains exactly one literal. An *assignment* σ is a set of literals such that $\sigma \cap \{\neg l \mid l \in \sigma\} = \emptyset$. By $var(\sigma)$ we denote the set $\{var(l) \mid l \in \sigma\}$. We sometimes interpret an assignment as a conjunction of its literals. An X-assignment σ is an assignment with $var(\sigma) \subseteq X$. If $var(\sigma) = X$, then σ is a *full* X-assignment, otherwise it is a *partial* X-assignment. We say that X-assignment τ is an *expansion* of X-assignment σ if $\sigma \subseteq \tau$.

If ϕ is a propositional formula and σ an assignment, then ϕ_σ denotes the formula obtained by setting the variable x to true if $x \in \sigma$, by setting x to false if $\neg x \in \sigma$, and by performing standard simplifications. A QBF $\forall x \Pi.\phi$ (resp. $\exists x \Pi.\phi$) is true if $\Pi.\phi_{\{x\}}$ and $\Pi.\phi_{\{\neg x\}}$ are true (resp. if $\Pi.\phi_{\{x\}}$ or $\Pi.\phi_{\{\neg x\}}$ is true). For example, the QBF $\exists x \forall y.(x \leftrightarrow y)$ is false and the QBF $\forall x \exists y.(x \leftrightarrow y)$ is true. *(Counter)-models* of QBFs can be described as sets of Boolean functions. Given a true QBF φ, a *model* of φ is a set of functions F such that for each existential variable $x \in var(\varphi)$, there is a function $f_x(y_1, \ldots, y_n) \in F$ with $y_i < x$ and y_i is universal. Further, the propositional formula φ_F which is obtained by replacing all its existential variables x by function $f_x \in F$ is valid iff F is a model. The functions of a model reflect the dependencies of the variables. If the first quantifier block is existential, then the functions of these variables are truth constants, i.e., assignments as introduced above. We are interested in those assignments.

Definition 1. *Let $\varphi = \exists X \Pi.\phi$ be a true QBF and let σ be a partial X-assignment. We call σ a* satisfying partial X-assignment *if the QBF $\forall X' \Pi.\phi_\sigma$ with $X' = X \setminus var(\sigma)$ is true (note that variables $X \setminus var(\sigma)$ are now universal). Then σ is also called* partial X-model *or* level-1 solution *of φ.*

Based on this definition every expansion of a partial X-model σ to a full X-assignment is an X-model. Hence, all variables of X not mentioned in σ may be set arbitrarily and preserve the satisfiability of the formula.

Example 1. Consider QBF $\exists x_1, x_2 \forall a \exists y.((x_1 \vee \neg x_2 \vee y) \wedge (x_1 \vee \neg x_2 \vee a \vee \neg y))$ with $X = \{x_1, x_2\}$. Then the X-assignment $\{x_1\}$ is a partial X-model, because both full X-assignments $\{x_1, x_2\}$ and $\{x_1, \neg x_2\}$ are X-models. In contrast, $\{\neg x_1\}$ is not a partial X-model, because $\{\neg x_1, x_2\}$ is not an X-model.

A *counter-model* of a false QBF φ is defined dually: a counter-model is a set of functions F such that for each universal variable x there is a function $f_x(y_1, \ldots, y_n) \in F$ with y_i existential and $y_i < x$. Further, φ_F, which is obtained by replacing all universal variables x by f_x, is unsatisfiable. The functions of the outermost universal variables are constants.

Definition 2. *Let $\varphi = \forall X \Pi.\phi$ be a false QBF and let σ be a partial X-assignment. We call σ a* falsifying partial X-assignment *if QBF $\exists X' \Pi.\phi_\sigma$ with $X' = X \setminus var(\sigma)$ is false. Then σ is also called a* partial X-counter-model *or* level-1 solution *of φ.*

Note that the term *solution* is used for models and counter-models.

4 Counting Models

Given a true QBF $\exists X \Pi.\phi$, we are interested in the number of full X-assignments σ such that $\Pi.\phi_\sigma$ is true. To this end, we enrich the formula with so-called *blocking clauses* until the formula becomes false. From the number of blocking clauses, we can infer the number of satisfying X-assignments, i.e., the number of X-models.

Definition 3 (Blocking Clause). *Let $\exists X \Pi.\phi$ be a true QBF and let σ be a partial X-model of this QBF. Then $\neg\sigma$ is a* blocking clause *of $\exists X \Pi.\phi$.*

By enriching the formula with blocking clauses, we exclude models, i.e., we avoid the same models to be discovered again when evaluating the enriched formula. The following lemma shows that the addition of blocking clauses obtained from a set M of partial X-models eliminates all partial X-models that are expansions of the X-models from M.

Lemma 1. *Let $\varphi = \exists X \Pi.\phi$ be a true QBF and let M be a set of partial X-models of φ. Then there is no partial X-model τ of $\varphi' = \exists X \Pi.(\phi \wedge \bigwedge_{\sigma \in M} \neg\sigma)$ with $\sigma \subseteq \tau$ for all $\sigma \in M$.*

Proof. Let τ be an X-assignment with $\sigma \subseteq \tau$ for some $\sigma \in M$. Then τ cannot be an X-model of φ', because τ falsifies the clause $\neg\sigma$. Hence, τ also falsifies φ'.

Based on blocking clauses, we can realize an enumerative approach to model counting. To this end, we need the following criterion to decide when all X-models have been covered.

Lemma 2. *Let $\varphi = \exists X \Pi.\phi$ be a true QBF and let M be a set of partial X-models. For each full X-model τ of φ, there exists a $\sigma \in M$ with $\sigma \subseteq \tau$ iff the QBF $\varphi' = \exists X \Pi.(\phi \wedge \bigwedge_{\sigma \in M} \neg\sigma)$ is false.*

Proof. We prove both directions by contradictions.
$\Rightarrow$: Assume there is an X-model τ of φ'. Then τ is also an X-model of φ. Since there exists an X-model $\sigma \in M$ with $\sigma \subseteq \tau$, τ cannot be an X-model of φ' by Lemma 1.
$\Leftarrow$: Assume there is an X-model τ of $\Pi.\phi$ such that there is no $\rho \in M$ with $\tau \subseteq \rho$. Then τ is an X-model of $\Pi.(\phi \wedge \bigwedge_{\sigma \in M} \neg\sigma)$ which is false by assumption. □

Corollary 1. *Let $\varphi = \exists X \Pi.\phi$ be a true QBF, let M be a set of full X-models with $\exists X \Pi.(\phi \wedge \bigwedge_{\sigma \in M} \neg\sigma)$ be false. Then φ has $|M|$ X-models.*

Based on this corollary, we could already build an enumeration-based model counter for QBF. A partial X-model describes a set of full X-models and yields a potentially exponentially more compact encoding leading to the following proposition.

Proposition 1. *Let $\varphi = \exists X \Pi.\phi$ be a true QBF and let $M = \{\sigma_1, \ldots, \sigma_n\}$ be a set of partial X-models of φ such that*

1. *$\exists X \Pi.(\phi \wedge \bigwedge_{\sigma \in M} \neg\sigma)$ is false and*
2. *$\sigma_k \in M$ is a partial X-model of $\exists X \Pi.(\phi \wedge \bigwedge_{i=1}^{k-1} \neg\sigma_i)$*

Then the number of X-models of φ is $\Sigma_{\sigma \in M} 2^{|X|-|\sigma|}$.

Proof. Because of the first condition, Lemma 2 applies. Therefore, all full X-models of φ can be obtained by expanding the elements of M. We further need to argue that no full X-model can be obtained by expanding several elements of M. Because of the second condition which imposes an order on the elements of M and the successive application of Lemma 1 it follows that each full X-model is only considered once. As one partial X-model σ of size $|\sigma|$ can be expanded to $2^{|X|-|\sigma|}$ full X-models, we get a total count of full X-models as stated above. □

Example 2. Consider QBF $\exists x_1, x_2 \forall a \exists y.((x_1 \vee \neg x_2 \vee y) \wedge (x_1 \vee \neg x_2 \vee a \vee \neg y))$ with $X = \{x_1, x_2\}$. Furthermore, let $M = \{\{x_1\}, \{\neg x_1, \neg x_2\}\}$, i.e., a set with one partial X-model and a full X-model such that $\{\neg x_1, \neg x_2\}\}$ is an X-model of $\exists x_1, x_2 \forall a \exists y.((x_1 \vee \neg x_2 \vee y) \wedge (x_1 \vee \neg x_2 \vee a \vee \neg y) \wedge \neg x_1)$.

As the QBF $\exists x_1, x_2 \forall a \exists y.((x_1 \vee \neg x_2 \vee y) \wedge (x_1 \vee \neg x_2 \vee a \vee \neg y) \wedge \neg x_1 \wedge (x_1 \vee x_2))$ is false, the formula has *three* $(2 + 1)$ X-models.

5 Counting Counter-Models

Dually to counting the X-models of a true QBF $\exists X \Pi.\phi$, one can also count the Y-counter-models (falsifying Y-assignments) of a false QBF $\forall Y \Pi.\psi$.

Example 3. The QBF $\forall y_1, y_2 \exists x.(y_1 \vee y_2 \vee x) \wedge (\neg y_1 \vee x) \wedge (\neg x)$ has partial Y-counter-models $\{\neg y_1, \neg y_2\}$ and $\{y_1\}$. The latter can be expanded to full Y-counter-models $\{y_1, y_2\}$ and $\{y_1, \neg y_2\}$.

To count such Y-counter-models we introduce the notion of *blocking cube* as the dual of the notion of blocking clause.

Definition 4 (Blocking Cube). *Let $\forall Y \Pi.\psi$ be a false QBF and let σ be a partial Y-counter-model of this QBF. Then σ is a* blocking cube *of $\forall Y \Pi.\psi$.*

Blocking cubes of the example above are $(\neg y_1 \wedge \neg y_2)$ and (y_1) as well as $(y_1 \wedge y_2)$ and $(y_1 \wedge \neg y_2)$. A blocking cube is a falsifying Y-assignment that we will use to extend a formula such that the same (super-)counter-models are excluded from the set of counter-models of a given formula. Therefore, we have to disjunctively add blocking cubes that have the following properties (as the proofs are similar to the proofs in the previous section, we omit them). In the same way as *partial* models allowed us to get shorter blocking clauses, partial counter-models will help us to get shorter blocking cubes, exponentially decreasing the number of required blocking cubes.

Lemma 3. *Let $\varphi = \forall Y \Pi.\psi$ be a false QBF and let M be a set of partial Y-counter-models of φ. Then there is no partial Y-counter-model ρ of $\varphi' = \forall Y \Pi.(\psi \vee \bigvee_{\sigma \in M} \sigma)$ with $\sigma \subseteq \rho$ for all $\sigma \in M$.*

This lemma ensures that all counter-models of the original formula that can be obtained by expanding a blocking cube are excluded from the set of counter-models of the formula enriched with blocking cubes. The next lemma states that if we disjunctively add all its Y-counter-models to a false QBF, it becomes true.

Lemma 4. *Let $\varphi = \forall Y \Pi.\psi$ be a false QBF and let M be a set of partial Y-counter-models. For each full Y-counter-model ρ of φ, there exists a $\sigma \in M$ with $\sigma \subseteq \rho$ iff the QBF $\varphi' = \forall Y \Pi.(\psi \vee \bigvee_{\sigma \in M} \sigma)$ is true.*

Now we can find the number of Y-counter-models of a false QBF as follows.

Proposition 2. *Let $\varphi = \forall Y \Pi.\psi$ be a false QBF and let $M = \{\sigma_1, \ldots, \sigma_n\}$ be a set of partial Y-counter-models of φ such that*

1. *$\forall Y \Pi.(\psi \vee \bigvee_{\sigma \in M} \sigma)$ is true and*
2. *$\sigma_k \in M$ is a partial Y-counter-model of $X\Pi.(\psi \vee \bigvee_{i=1}^{k-1} \sigma_i)$*

Then the number of Y-counter-models of $\Pi.\psi$ is $\Sigma_{\sigma \in M} 2^{|Y|-|\sigma|}$.

Example 4. Given false QBF $\varphi = \forall y_1, y_2 \exists x.(y_1 \vee y_2 \vee x) \wedge (\neg y_1 \vee x) \wedge (\neg x)$ with partial Y-counter-models $\{\neg y_1, \neg y_2\}$ and $\{y_1\}$. As the QBF

$$\varphi' = \forall y_1, y_2 \exists x.((y_1 \vee y_2 \vee x) \wedge (\neg y_1 \vee x) \wedge (\neg x)) \vee (\neg y_1 \wedge \neg y_2) \vee (y_1)$$

is true, we can conclude that φ has $1 + 2$ counter-models.

Note that in contrast to counting X-models, we lose the PCNF structure when counting Y-counter-models. To obtain a PCNF again, we perform the well-known Plaisted-Greenbaum [31] transformation which introduces additional variables. The following lemma shows that this transformation does not change the number of (counter)-models.

Lemma 5. *Let $\forall Y \Pi.\psi$ be a false QBF and σ be a Y-counter-model of $\Pi.\psi$. Then QBF*

$$\varphi_1 = \Pi \exists t_1, t_2.((t_1 \rightarrow \psi) \wedge (t_2 \rightarrow \sigma) \wedge (t_1 \vee t_2))$$

has the same number of Y-counter-models as the QBF $\varphi_2 = \Pi.(\psi \vee \sigma)$.

Proof. We show that φ_1 and φ_2 have the same Y-counter-models.

$\Rightarrow$: Let τ be a Y-counter-model of φ_1. The subformula σ which contains only literals from Y has to evaluate to false under τ, because otherwise t_2 would become a pure literal in clause $(t_1 \vee t_2)$, in consequence $\neg t_1$ would become a pure literal as well, and then φ_1 would evaluate to true under τ. Hence, σ has to be false under τ. Then $\neg t_2$ becomes a unit clause, and by propagation t_1 becomes a unit clause as well, simplifying φ_1 under τ to $\Pi.(\psi_\tau)$ which has to be false (because τ is an Y-counter-model). It follows that τ is also a Y-counter-model of φ_2.

$\Leftarrow$: Let τ be a Y-counter-model of φ_2. Then σ is false under τ and so is $\Pi.\psi_\tau$. It follows that τ is a Y-counter-model of φ_1 as well.

It is easy to see that the lemma above directly transfers to extending a QBF with m Y-counter-models by introducing $m + 1$ new variables.

Example 5. Applying the Plaisted-Greenbaum transformation on QBF φ of Example 4 results in $\forall y_1, y_2 \exists x, t_1, t2, t_3.(t_1 \rightarrow (y_1 \vee y_2 \vee x) \wedge (\neg y_1 \vee x) \wedge (\neg x)) \wedge (t_2 \rightarrow (\neg y_1 \wedge \neg y_2)) \wedge (t_3 \rightarrow (y_1)) \wedge (t_1 \vee t_2 \vee t_3)$ which can efficiently be transformed into PCNF by standard logical rules.

input : QBF $\Phi = \mathsf{Q}Z\Pi.\phi$
output: Numbers of Z-Solutions of Φ

$c \leftarrow 0$; $i \leftarrow 0$;
DepQBF.init ($\mathsf{Q}Z\Pi\exists t_0.(t_0 \rightarrow \phi)$);
DepQBF.assume (t_0);
$(v, \sigma) \leftarrow$ DepQBF.solve() ;
if $(v = \top \ \& \ \mathsf{Q} = \forall) \| (v = \bot \ \& \ \mathsf{Q} = \exists)$ **then**
  return -1;
end
if $v = \top$ **then**
  DepQBF.add ((t_0));
end
do
  $c \leftarrow c + 2^{|Z|-|\sigma|}$;
  if $v = \top$ **then**
    DepQBF.add ($\neg\sigma$);
  end
  else
    $i++$;
    DepQBF.add ($t_i \rightarrow \sigma$);
    DepQBF.assume (($t_0 \vee \ldots \vee t_i$));
  end
  $(v, \sigma) \leftarrow$ DepQBF.solve() ;
while $(v = \bot \ \& \ \mathsf{Q} = \forall) \| (v = \top \ \& \ \mathsf{Q} = \exists)$;
return c;

Algorithm 1: OuterCount (Φ)

6 Implementation

To implement the previously presented approach we use the incremental solving interface of the search-based QBF solver DepQBF [24] to enrich the formula with blocking clauses or blocking cubes. In the case of true formulas, in each solver call, the formula is enriched with additional clauses. In the case of false formulas, it is necessary to update a clause. For this purpose, we use the push and pop functions of DepQBF which allows us to add a clause that is only available for one solver run: we push a clause to DepQBF, run DepQBF and evaluate the result. If another run is necessary we pop the clause and push the updated clause to DepQBF. This is necessary to increase the disjunction of the definitions introduced by the Plaisted-Greenbaum transformation in the case of counter-model counting.

Algorithm 1 takes as input a QBF Φ starting with quantifier block $\mathsf{Q}Z$ for which the number of Z-solutions must be determined. To this end, we initialize the solver with the QBF $\mathsf{Q}Z\Pi\exists t_0.(t_0 \rightarrow \phi)$ where t_0 is a fresh existential variable that is added to the innermost scope. During the first solver call, t_0 is assumed to be true hence it does not have any effect on the solving result. If Q is $\exists$ (resp., $\forall$) and Φ is false (resp., true), then -1 is returned as there is nothing

Table 1. Formulas for which the exact count could be found without preprocessing. With preprocessing, more formulas run into timeouts (TO).

		without preprocessing					with preprocessing				
	instance	Q	# bl	# v	# s	t(s)	Q	# bl	# v	# s	t(s)
true formulas	gttt_1_1...torus_b	∃	17	355	1	53	∀	16	–	–	–
	gttt_2_2...torus_b	∃	9	810	15	103	∃	9	31	15	31
	gttt_2_2..._b	∃	9	754	42	1991	∃	9	37	42	192
	k_ph_n-11	∃	5	3	1	0.2	∃	1	110	> 1K	TO
	k_ph_n-15	∃	5	3	1	1.3	∃	1	210	> 1K	TO
	k_ph_n-18	∃	5	3	1	40	∃	3	454	> 2K	TO
	k_ph_n-19	∃	5	3	1	6.14	∃	3	507	> 2K	TO
	k_ph_n-20	∃	5	3	1	9.72	∃	3	564	> 700	TO
false formulas	arbiter-05-...-depth-8	∀	18	10	1	0.1	∀	14	10	214	24.5
	arbiter-06-...-depth-11	∀	24	12	1	6.5	∀	20	12	342	679
	arbiter-06-...-depth-15	∀	32	12	1	187	∀	28	12	> 40	TO
	arbiter-09-...-depth-15	∀	32	18	1	523	∀	28	12	> 500	TO
	W4...tbm_05..8S...1	∀	20	10	10	60	∀	18	10	10	1023
	W4...tbm_25..7S...3	∀	20	10	10	520	∀	18	10	> 8	TO
	W4...tbm_26..7S...3	∀	20	10	9	689	∀	18	9	> 4	TO

Q . . . quantifier type at level 1 #bl . . . number of quantifier blocks
#v . . . size of first quantifier block
#s . . . number of solutions t(s) . . . runtime in seconds

to count. If Φ is true, t_0 is permanently added as a unit clause, because in this case it is not required for the normal form transformation. Next, the counting variable c is updated taking into account the size of the found solution. If Φ is true, $\neg\sigma$ is added, otherwise $t_i \rightarrow \sigma$ in the form of $|\sigma|$ binary clauses is added. Further, the clause $(t_0 \vee \ldots \vee t_i)$ is assumed for disjunctively excluding the found solutions. The algorithm iterates until the formula becomes false (resp. true) and, in that case, returns c, the number of Z-solutions. Our tool (together with the experimental results of the evaluation described below) is available at

`https://github.com/marseidl/outer-count`

7 Evaluation

As our tool is the first tool for practical QBF solution counting, there are no other tools to compare with. Further, no benchmarks are available. Therefore, we established two sets of benchmarks. First, we evaluated our tool on the formulas of the PCNF track of QBFEVAL 2020,[4] the most recent QBF competition. In a prerun, we identified 68 true formulas that start with an existential quantifier block, and 34 false formulas that start with a universal quantifier block solvable

[4] `http://www.qbfeval.org`

by plain DepQBF. We also applied the preprocessor bloqqer[5] to obtain a second set of formulas. Amongst other techniques, bloqqer performs existential variable elimination, universal expansion, blocked clause elimination and equality reasoning. More details on QBF preprocessing and bloqqer can be found in [18]. The preprocessor bloqqer directly solved seven true formulas and eight false formulas, while for 9 true formulas and 1 false formulas of the preprocessed benchmarks no solution was found at all. These formulas were excluded from the benchmark set, leading to a set of 52 true formulas and 25 false formulas. All experiments were carried out on a machine with 128 AMD EPYC 7501 processors. For each formula, we limited the time to 3600 seconds and the memory to 8GB.

Our tool could determine the exact level-1 solution count for eight true formulas and for seven false formulas. Details on those formulas are shown in the left part of Table 1. The eight true formulas contain three formulas from encodings of a generalization of the two-player-game TIC-TAC-TOE (gttt*) to synthesize a winning strategy for one player [12] and five formulas (k_ph*) of QBF encodings to solve formulas from Modal Logics [30]. While the formulas of the gttt* family have a very deep quantifier structure (up to 17 quantifier blocks) and many variables in the first quantifier block (up to more than 800 variables), the k_ph* formulas have only five quantifier blocks and three variables in the first quantifier block. For the gttt* formulas one, 15, and 42 level-1 solutions were found, the k_ph* formulas have exactly one level-1 solution each. Preprocessing (right part of Table 1) considerably changed the structure of the formulas and also their solution space to some extent. For formula gttt_1_1* the variables of the first existential quantifier block were eliminated, resulting in a formula starting with a universal quantifier block. Hence it does not have any level-1 solutions anymore. For the other two gttt* formulas, the number of variables decreased by preprocessing, but the number of level-1 solutions did not change. This might be an indication that there is potential to optimize the encoding. For the k_ph* formulas the number of quantifier blocks decreased from five to three or even to one. For these formulas, the exact number of level-1 solutions could not be determined within one hour. We found more than 700, 1.200 (twice), 2.000, and 2.500 solutions for the formulas of this family.

The seven false formulas, for which we could determine the exact level-1 solution count, stem from verification. Those formulas have a huge number of quantifier blocks (up to 32) but only few variables in the first quantifier block (up to 18). Also, the number of level-1 solutions is rather small (up to 10). This changes drastically, when preprocessing is enabled. For four formulas the exact level-1 solution count could not be determined, and for formulas with originally one level-1 the solution, several hundred were found (see right part of Table 1).

Figure 1 shows the number of level-1 solutions of all formulas counted within a time-frame of one hour, i.e., we get a lower bound for the level-1 solution count. We observe that in general, preprocessing increases the solution space. This could explain why solvers that do not rely on the formula structure can often solve preprocessed formulas more efficiently. Similarly, as reported for solving #SAT

[5] `http://fmv.jku.at/bloqqer`

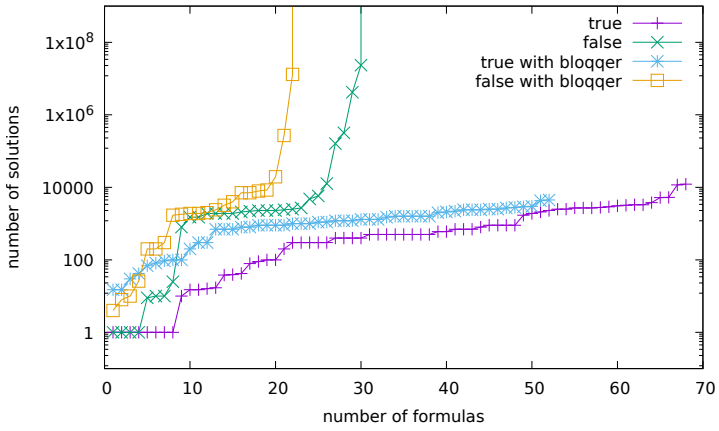

Fig. 1. Solution counts found within one hour and with 8GB memory limit.

problems, also in the context of QBF counting the number of solutions can get very large.

8 Conclusion

In this paper, we presented the first practical QBF model counting approach for the outermost (level-1) quantified variables. To this end, we have lifted the idea of enriching the formula with blocking clauses from SAT to counting the models of true QBFs. We also introduced the dual approach for counting counter-models of false formulas. We applied the incremental solving interface of the state-of-the-art QBF solver DepQBF to enrich the formula with blocking clauses (or blocking cubes), resulting in a very elegant implementation. Our empirical evaluation demonstrates that counting for QBFs is feasible to a certain extent (at least for getting lower bounds). This is particularly interesting to analyze the effects of preprocessing. In the future, we plan to apply solution minimization to exponentially reduce the search space. Short blocking clauses rule out a larger portion of the search space, hence model minimization techniques have been developed for other formalisms [1, 23, 20]. Further, counting could be directly integrated into a state-of-the-art QBF solver in a similar way it has been done for SAT model counters. The ultimate next step is to generalize our approach for counting functions to perform level-2 counting and beyond.

References

1. Aziz, R.A., Chu, G., Muise, C.J., Stuckey, P.J.: #∃SAT: Projected model counting. In: SAT. LNCS, vol. 9340, pp. 121–137. Springer (2015)

2. Bailey, J., Stuckey, P.J.: Discovery of minimal unsatisfiable subsets of constraints using hitting set dualization. In: PADL. LNCS, vol. 3350, pp. 174–186. Springer (2005)
3. Baluta, T., Shen, S., Shinde, S., Meel, K.S., Saxena, P.: Quantitative verification of neural networks and its security applications. In: CCS. pp. 1249–1264. ACM (2019)
4. Bayardo Jr., R., Pehoushek, J.D.: Counting models using connected components. In: AAAI/IAAI. pp. 157–162. AAAI Press / The MIT Press (2000)
5. Becker, B., Ehlers, R., Lewis, M., Marin, P.: ALLQBF solving by computational learning. In: ATVA. LNCS, vol. 7561, pp. 370–384. Springer (2012)
6. Bendík, J., Cerná, I.: Replication-guided enumeration of minimal unsatisfiable subsets. In: CP. LNCS, vol. 12333, pp. 37–54. Springer (2020)
7. Bendík, J., Cerna, I.: Rotation based MSS/MCS enumeration. In: LPAR. EPIC, vol. 73, pp. 120–137. EasyChair (2020)
8. Biondi, F., Enescu, M.A., Heuser, A., Legay, A., Meel, K.S., Quilbeuf, J.: Scalable approximation of quantitative information flow in programs. In: VMCAI. LNCS, vol. 10747, pp. 71–93. Springer (2018)
9. Birnbaum, E., Lozinskii, E.L.: The good old Davis-Putnam procedure helps counting models. J. Artif. Intell. Res. **10**, 457–477 (1999)
10. Chakraborty, S., Meel, K.S., Vardi, M.Y.: Algorithmic improvements in approximate counting for probabilistic inference: From linear to logarithmic SAT calls. In: IJCAI. pp. 3569–3576. IJCAI/AAAI Press (2016)
11. Chakraborty, S., Meel, K.S., Vardi, M.Y.: Chapter 26. Approximate model counting. In: Handbook of Satisfiability, Frontiers in Artificial Intelligence and Applications, vol. 336, pp. 1015–1045. IOS Press (2021)
12. Diptarama, Yoshinaka, R., Shinohara, A.: QBF encoding of generalized Tic-Tac-Toe. In: QBF@SAT. CEUR Workshop Proceedings, vol. 1719, pp. 14–26. CEUR-WS.org (2016)
13. Dubois, O.: Counting the number of solutions for instances of satisfiability. Theor. Comput. Sci. **81**(1), 49–64 (1991)
14. Gebser, M., Kaufmann, B., Neumann, A., Schaub, T.: Conflict-driven answer set enumeration. In: LPNMR. LNCS, vol. 4483, pp. 136–148. Springer (2007)
15. Giunchiglia, E., Narizzano, M., Tacchella, A.: Learning for quantified Boolean logic satisfiability. In: AAAI/IAAI. pp. 649–654. AAAI Press / The MIT Press (2002)
16. Gomes, C.P., Sabharwal, A., Selman, B.: Chapter 25. Model counting. In: Handbook of Satisfiability, Frontiers in Artificial Intelligence and Applications, vol. 336, pp. 993–1014. IOS Press, Amsterdam, Netherlands (Apr 2021)
17. Hemaspaandra, L.A., Vollmer, H.: The satanic notations: counting classes beyond #P and other definitional adventures. SIGACT News **26**(1), 2–13 (1995)
18. Heule, M., Järvisalo, M., Lonsing, F., Seidl, M., Biere, A.: Clause elimination for SAT and QSAT. J. Artif. Intell. Res. **53**, 127–168 (2015)
19. Hunter, A., Konieczny, S.: Measuring inconsistency through minimal inconsistent sets. In: KR. pp. 358–366. AAAI Press (2008)
20. Jin, H., Han, H., Somenzi, F.: Efficient conflict analysis for finding all satisfying assignments of a Boolean circuit. In: TACAS. LNCS, vol. 3440, pp. 287–300. Springer (2005)
21. Ladner, R.E.: Polynomial space counting problems. SIAM J. Comput. **18**(6), 1087–1097 (1989)
22. Letz, R.: Lemma and model caching in decision procedures for quantified Boolean formulas. In: TABLEAUX 2002. LNCS, vol. 2381, pp. 160–175. Springer (2002)

23. Li, B., Hsiao, M.S., Sheng, S.: A novel SAT all-solutions solver for efficient preimage computation. In: DATE. pp. 272–279. IEEE Computer Society (2004)
24. 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)
25. McAreavey, K., Liu, W., Miller, P.: Computational approaches to finding and measuring inconsistency in arbitrary knowledge bases. Int. J. Approx. Reason. **55**(8), 1659–1693 (2014)
26. Möhle, S., Biere, A.: Combining conflict-driven clause learning and chronological backtracking for propositional model counting. In: GCAI. EPIC, vol. 65, pp. 113–126. EasyChair (2019)
27. Morgado, A., Liffiton, M.H., Marques-Silva, J.: MaxSAT-based MCS enumeration. In: HVC. LNCS, vol. 7857, pp. 86–101. Springer (2012)
28. Narodytska, N., Bjørner, N., Marinescu, M.V., Sagiv, M.: Core-guided minimal correction set and core enumeration. In: IJCAI. pp. 1353–1361. ijcai.org (2018)
29. Narodytska, N., Shrotri, A.A., Meel, K.S., Ignatiev, A., Marques-Silva, J.: Assessing heuristic machine learning explanations with model counting. In: SAT. LNCS, vol. 11628, pp. 267–278. Springer (2019)
30. Pan, G., Vardi, M.Y.: Symbolic decision procedures for QBF. In: CP. LNCS, vol. 3258, pp. 453–467. Springer (2004)
31. Plaisted, D.A., Greenbaum, S.: A structure-preserving clause form translation. J. Symb. Comput. **2**(3), 293–304 (1986)
32. Pulina, L., Seidl, M.: The 2016 and 2017 QBF solvers evaluations (QBFEVAL'16 and QBFEVAL'17). Artif. Intell. **274**, 224–248 (2019)
33. Sang, T., Beame, P., Kautz, H.A.: Performing Bayesian inference by weighted model counting. In: AAAI. pp. 475–482. AAAI Press / The MIT Press (2005)
34. Shukla, A., Biere, A., Pulina, L., Seidl, M.: A survey on applications of quantified Boolean formulas. In: ICTAI. pp. 78–84. IEEE (2019)
35. Stockmeyer, L.J., Meyer, A.R.: Word problems requiring exponential time: Preliminary report. In: STOC. pp. 1–9. ACM (1973)
36. Thimm, M., Wallner, J.P.: On the complexity of inconsistency measurement. Artif. Intell. **275**, 411–456 (2019)
37. Zhang, L., Malik, S.: Towards a symmetric treatment of satisfaction and conflicts in quantified Boolean formula evaluation. In: CP. Lecture Notes in Computer Science, vol. 2470, pp. 200–215. Springer (2002)
38. Zhou, Z., Qian, Z., Reiter, M.K., Zhang, Y.: Static evaluation of noninterference using approximate model counting. In: SP. pp. 514–528. IEEE Computer Society (2018)