A Top-Down Tree Model Counter for Quantified Boolean Formulas

Florent Capelli[1] , **Jean-Marie Lagniez**[1] , **Andreas Plank**[2] and **Martina Seidl**[2]
[1]Univ. Artois, CNRS, Centre de Recherche en Informatique de Lens (CRIL), F-62300 Lens, France
[2]Institute for Symbolic Artificial Intelligence, JKU Linz, Austria
{capelli, lagniez}@cril.fr, {andreas.plank, martina.seidl}@jku.at

Abstract

This paper addresses the challenge of solution counting for Quantified Boolean Formulas (QBFs), a task distinct from the well-established model counting problem for SAT (#SAT). Unlike SAT, where models are straightforward assignments to Boolean variables, QBF solution counting involves tree models that capture dependencies among variables within different quantifier blocks. We present a comprehensive top-down tree model counter capable of handling diverse satisfiable QBFs. Highlighting the pivotal role of the branching heuristic, which needs to consider variables in accordance with quantification blocks, we further underscore the significance of dealing with connected components, free variables, and caching. Experimental results indicate that our proposed approach for counting tree models of QBF formulas is highly efficient in practice, surpassing existing state-of-the-art methods designed for this specific purpose.

1 Introduction

Problems of the polynomial hierarchy [Stockmeyer, 1976] emerge in a wide range of applications from artificial intelligence, including reasoning tasks in planning, argumentation, two-player games, formal verification, and synthesis to name a few examples. To handle such problems in a uniform manner, *quantified Boolean formulas* (QBFs) offer an appealing framework [Shukla *et al.*, 2019] with advanced theory and practice of solving technology [Beyersdorff *et al.*, 2021; Kleine Büning and Bubeck, 2021; Giunchiglia *et al.*, 2021].

QBFs, extending propositional logic (SAT) by introducing quantifiers over Boolean variables, exhibit notable similarities with SAT. However, the incorporation of universal quantifiers introduces fundamental distinctions. Unlike SAT, a solution to a true QBF is not merely an assignment ensuring the formula's satisfiability. Instead, it takes the form of a tree structure, delineating, for all combinations of universal variable assignments, how existential variables must be set to make the formula evaluate to true. This inherent complexity renders the evaluation of QBFs more intricate compared to SAT. Example 1 provides a concrete illustration of this tree structure.

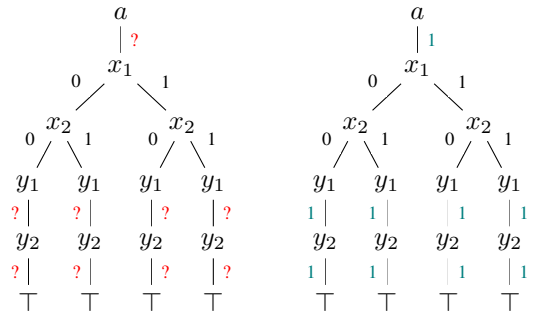

Figure 1: Tree model structure (left) and concrete model of QBF (1)

Example 1. *Consider the QBF* Φ *given by:*

$$\exists a \forall x_1 x_2 \exists y_1 y_2 . (\neg a \vee x_1 \vee y_1) \wedge (\neg a \vee x_2 \vee y_2) \wedge (a \vee x_2 \vee y_2) \quad (1)$$

This formula has tree models of the structure as shown on the left of Figure 1. On the right, we see a concrete instance of a tree model with concrete values for the existential variables.

The task of a QBF solver is to determine values for the existential variables in a way that each path of the tree encompasses an assignment satisfying the propositional part of the QBF or to demonstrate that no such assignment exists. In the case of the previous example, there exist 80 distinct tree models, signifying that Φ has 80 solutions. These tree models encapsulate crucial information pertaining to the solution of the original reasoning task, such as the plan of a planning problem, an error trace in a verification problem, or the synthesized program in formal synthesis. Consequently, extensive research has been conducted on extracting solutions from QBF solvers [Balabanov and Jiang, 2012; Balabanov *et al.*, 2015; Beyersdorff *et al.*, 2014; Goultiaeva *et al.*, 2011; Chew and Slivovsky, 2022].

In contrast to the extensive exploration of extracting and analyzing solutions in QBFs, the problem of counting the number of solutions, also known as #QBF, has predominantly received theoretical attention [Ladner, 1989; Hemaspaandra and Vollmer, 1995; Bauland *et al.*, 2005]. Some work has delved into counting solutions for QBFs within specific constraints. For instance, in [Shukla *et al.*, 2022], an approach to counting solutions at the outer level was introduced, focusing on a true QBF with the first quantifier block $\exists X$ being

existentially quantified. This method enumerates the number of Boolean assignments to variables in X that render the QBF true [Birnbaum and Lozinskii, 1999; Dubois, 1991; Becker *et al.*, 2012]. The enumerative strategy has been extended to the second level, enabling the counting of solutions for true QBFs in the form of $\forall X \exists Y$ [Plank *et al.*, 2023]. To achieve this, Skolem functions representing tree models as circuits are incorporated into the formula until the formula evaluates to false. An approximate approach in the related field of Boolean function synthesis will be presented in [Shaw *et al.*, 2024]. However, as of now, there is no approach available that can count solutions for arbitrary QBFs.

It is crucial to distinguish #QBF from classical counting tasks like #SAT, projected model counting [Aziz *et al.*, 2015], (weighted) Max#SAT [Fremont *et al.*, 2017; Audemard *et al.*, 2022], E-MajSAT [Littman *et al.*, 1998], and the stochastic satisfiability (SSAT) [Majercik, 2009] problems. This distinction is notable in the choice of semi-ring for aggregating sub-counts, and the special considerations required for handling connected components, caching, and free variables.

Contributions The principal contribution of this work lies in the development of what, to the best of our knowledge, is the first QBF model counter capable of addressing the counting problem for formulas with arbitrary quantifier prefixes. To this end, we lay the groundwork by establishing the required theoretical foundations and present an efficient implementation. Notably, our approach outperforms existing methods tailored for QBFs with just one quantifier alternation (2QBFs). Through empirical demonstrations, we showcase the prowess of our approach in determining solution counts for instances from diverse formula families, even those considered challenging for contemporary QBF solvers.

2 Preliminaries

We consider *quantified Boolean formulas* in the form $\Pi.\phi$, where $\Pi = Q_1 X_1 \dots Q_n X_n$ is referred to as the *quantifier prefix* (with $Q_i \in \{\forall, \exists\}$ and $Q_i \neq Q_{i+1}$ for $i \in \{1, \dots, n-1\}$, and $X_1, \dots, X_n$ are pairwise disjoint, non-empty sets of Boolean variables). We denote by $\Pi^{\exists}_{Y}$ the quantifier prefix derived from Π by removing existentially quantified variables that do not belong to Y. For instance, if $\Pi = \forall x_1 x_2 \exists y_1 y_2 \forall z_1 z_2 \exists y_3$, then $\Pi^{\exists}_{\{y_1\}} = \forall x_1 x_2 \exists y_1 \forall z_1 z_2$. Additionally, we use the notation $\Pi \setminus \{x\}$ to represent the quantifier prefix Π with the variable x removed. In the previous example, $\Pi \setminus \{x_2\} = \forall x_1 \exists y_1 y_2 \forall z_1 z_2 \exists y_3$.

The *matrix* ϕ is a propositional formula in conjunctive normal form (CNF) over variables X_i. The set $Var(\phi)$ represents the variables occurring in ϕ. A CNF consists of a conjunction of clauses. Each clause is a disjunction of literals. A literal ℓ is a variable x or the negation $\neg x$ of a variable x. If ℓ is a literal, then $Var(\ell) = x$ if $\ell = x$ or $\ell = \neg x$. Additionally, $\bar{\ell}$ is defined as x if $\ell = \neg x$ and $\bar{\ell} = \neg x$ otherwise.

An *assignment* μ of ϕ is a set of literals over (a subset of) $Var(\phi)$ such that there is no $\ell \in \mu$ with $\bar{\ell} \in \mu$. For an assignment μ, $Var(\mu) = \{\, Var(\ell) \mid \ell \in \mu \}$. If $Var(\mu) = Var(\phi)$, then μ is a *full* assignment, else it is a *partial* assignment. By 2^X we denote the set of all possible assignments to variables in X. Given a propositional formula ϕ and an assignment μ, then $\phi_{|\mu}$ denotes the formula obtained when setting all literals $x \in \mu$ to true and all literals $\neg x \in \mu$ to false, respectively. For setting a single variable x to true (resp. to false) in a CNF ϕ, we write $\phi_{|x}$ (resp. $\phi_{|\neg x}$). A QBF $\exists x \Pi.\phi$ is true iff $\Pi.\phi_{|x}$ or $\Pi.\phi_{|\neg x}$ is true. A QBF $\forall x \Pi.\phi$ is true iff $\Pi.\phi_{|x}$ and $\Pi.\phi_{|\neg x}$ are true. Because of this semantics, a prefix $\Pi = Q_1 X_1 \dots Q_n X_n$ of QBF $\Pi.\phi$ induces an ordering on the variables: $x_i <_\Pi x_j$ if $x_i \in X_i$, $x_j \in X_j$ and $i < j$.

A *model* for a QBF $\Phi = \Pi.\phi$ with $|\, Var(\Phi)| = m$ is a tree of height $m+1$ such that every node at level $k \in \{1, \dots, m\}$ is labeled with a variable x_k in the order of the prefix, i.e., if variable x_j is at level j and variable x_k is at level k with $j \leq k$ then $x_j \leq_\Pi x_k$. The order of the variables from the same quantifier block X can be chosen in an arbitrary manner. We consider two tree models that only differ in the order of variables from the same quantifier block as equal. A node at level k has one child if $x_k \in X_i$ with $Q_i = \exists$, and two children if $x_k \in X_i$ with $Q_i = \forall$. We denote the number of different QBF models of a QBF Φ by $\#(\Phi)$. For instance, for QBF Φ from Example 1, we observe that $\#(\Phi) = 80$.

Algorithm 1: d4-QBF

input : $\Phi = \Pi.\phi$: a QBF formula.
output: the number of tree models of Φ.

1 **if** ϕ *is unsat* **then return** 0;
2 $(\phi', \mu) \leftarrow \texttt{BCP}(\phi)$;
3 **if** $\exists Q_i \in \Pi$ *s.t.* $Q_i = \forall \wedge X_i \cap Var(\mu) \neq \emptyset$ **then return** 0;
4 **if** ϕ' *is valid* **then return** $\texttt{CountReduct}(\Pi, \emptyset, 1)$;
5 $\Pi' \leftarrow \texttt{Shrink}(\Pi,\, Var(\phi'))$;
6 **if** $cache[\phi'] \neq \emptyset$ **then return** $\texttt{CountReduct}(\Pi, \Pi', \phi')$;
7 $c \leftarrow 1$;
8 $\gamma \leftarrow \texttt{ConnectedComponent}(\phi')$;
9 **if** $|\gamma| > 1$ **then**
10 **for** $\phi'' \in \gamma$ **do**
11 $c \leftarrow c \times \texttt{d4-QBF}(\Pi'^{\exists}_{Var(\phi'')}.\phi'')$;
12 **else**
13 $v \leftarrow \texttt{selectVar}(X'_1)$;
14 $c_1 \leftarrow \texttt{d4-QBF}(\Pi' \setminus \{v\}.\phi'_{|v})$;
15 $c_2 \leftarrow \texttt{d4-QBF}(\Pi' \setminus \{v\}.\phi'_{|\neg v})$;
16 $c \leftarrow (Q'_1 = \forall)?c_1 \times c_2 : c_1 + c_2$;
17 $cache[\phi'] \leftarrow c$;
18 **return** $\texttt{CountReduct}(\Pi, \Pi', c)$;

3 A Top-Down QBF Counter

3.1 Algorithm

Below, we present our algorithm, referred to as d4-QBF, aimed at computing the count of tree models for a given QBF. The algorithm is depicted in Algorithm 1. Specifically, it commences by evaluating several base cases (lines 1–6). Initially, it checks if the formula ϕ, the matrix of the QBF Φ, is unsatisfiable by invoking a SAT solver. If the formula is unsatisfiable, then the number of tree models is 0, and the function returns this value (Line 1). Subsequently, the algorithm

computes the formula resulting from the Boolean constraint propagation (BCP) procedure by setting literals in clauses of size one to true. The results of BCP are stored in ϕ' and μ. Here, ϕ' represents the formula obtained after simplification, and μ signifies the set of unit literals (line 2). Importantly, $Var(\phi') \cap Var(\mu) = \emptyset$, and $\phi \equiv \phi' \wedge \mu$. This transformation does not change the number of tree models, which is an easy consequence of the definition of tree models:

Lemma 1. *Let ϕ and ψ be two CNFs on variables X that have the same satisfying assignments over X and let Π be a quantifier prefix s.t. $Var(\Pi) = X$. Then $\#(\Pi.\phi) = \#(\Pi.\psi)$.*

If a QBF contains a unit clause with a universal literal, then the QBF is false. Hence, if μ contains a universal literal, the QBF Φ is false, and the function returns 0 (line 3). If ϕ' is valid, the algorithm computes the number of tree models for Φ directly from Π. This is done using a subroutine `CacheReduct` whose correctness is proven in Corollary 1.

Finally, the algorithm checks whether ϕ', obtained at line 2, has been previously computed by inspecting the $cache$ structure. Notably, $cache$ is a global variable consisting of a hash map associating formulas with natural numbers. However, one has to be careful as it is possible that the same formula is computed twice with different prefixes, inducing different model count. To ensure cache consistency, we define in Section 3.4 a notion of canonical prefix obtained using function `Shrink`. Inuitively, `Shrink` removes existential variables that are not in $Var(\phi')$ and some universal quantified variables as long as they remain on the first quantification block. We give more details in Section 3.4 where we are able to prove that whenever a cache hit occurs, then $cache[\phi']$ contains the value $\#(\Pi'.\phi')$ where Π' is obtained from Π using `Shrink`. The function `CountReduct` then recovers $\#(\Pi.\phi')$ from this value. Its correctness is proven in Lemma 3.

After addressing all base cases, the algorithm proceeds to the recursive case. The variable c, designated for storing the model count, is initialized to 1 (line 7). Subsequently, the connected components of ϕ are computed in γ utilizing the `ConnectedComponent` function applied to ϕ'. This function yields syntactically disjoint sub-formulas of ϕ' (line 8). As emphasized in Subsection 3.3, this step plays a pivotal role in decomposing the formula's complexity, thereby enhancing the efficiency of the tree model count computation.

Then, once the connected component γ has been computed, we consider two cases based on the size of γ. If $|\gamma|$ is greater than 1 (line 9), Proposition 1 becomes instrumental. It allows us to compute the tree model count for each sub-formula independently by invoking `d4-QBF` recursively and then multiplying the results (lines 10–11). To effectively leverage Proposition 1, we adjust the quantifier prefix to include only the existentially quantified variables that are necessary (i.e. $\Pi'^{\exists}_{Var(\phi')}$, see Subsection 3.3 for more details).

In the scenario where the formula contains only one connected component, a decision variable v is chosen (line 13) among the variables of the first quantifier block using the `selectVar` function. Subsequently, two recursive calls to `d4-QBF` are made, corresponding to the two ways of conditioning v in $\Pi'.\phi'$ (lines 14–15). As elucidated in Theorem 1 (refer to Subsection 3.2 for a detailed explanation of the branching strategy), the synthesis of results involves multiplication when the variable v is universally quantified and addition when v is existentially quantified (see line 16).

Finally, the cache is updated with the computed count associated with $\Pi'.\phi'$ (line 17). Since the quantification order remains constant throughout the search, using ϕ' as the key instead of $\Pi'.\phi'$ suffices as this QBF lacks free variables. The correctness of this approach is proven in Section 3.4. Once ϕ' is added to the cache, the `CountReduct` function can be invoked to obtain the correct tree model count (line 18).

3.2 Branching Strategy

Our algorithm operates recursively by progressing through variable branching in the formula and evaluating the resulting sub-formula. As emphasized earlier, accurate counting of tree models requires adherence to the order of variables outlined by the quantifier prefix. In addition to this requirement, it is crucial to compute the precise number of tree models by appropriately aggregating the sub-counts obtained. The subsequent theorem illustrates the method for recombining subcounts based on the type of quantifier used for branching.

Theorem 1. *Let $\Phi = Qx.\Psi$. Then $\#(\Phi) = \#(\Psi_{|x}) \times \#(\Psi_{|\neg x})$ if $Q = \forall$ and $\#(\Phi) = \#(\Psi_{|x}) + \#(\Psi_{|\neg x})$ if $Q = \exists$.*

Proof. First assume that $Q = \forall$. By definition, the models of Φ are the set of trees whose root is labeled by x and is connected to two subtrees T_0 and T_1 such that T_0 is a model of $\Psi_{|\neg x}$ and T_1 is a model of $\Psi_{|x}$. There are obviously $\#(\Psi_{|x}) \times \#(\Psi_{|\neg x})$ such trees hence $\#(\Phi) = \#(\Psi_{|x}) \times \#(\Psi_{|\neg x})$.

Now assume $Q = \exists$. By definition, the models of Φ are the set of trees whose root r is labeled by x. Further, its root r is connected to a subtree T with root r' such that: if the edge (r, r') is labeled by 0 then T is a model of $\Psi_{|\neg x}$ and if it is labeled by 1 then T is a model of $\Psi_{|x}$. Hence, there are $\#(\Psi_{|\neg x})$ models of Φ such that (r, r') is labeled by 0 and $\#(\Psi_{|x})$ models of Φ such that (r, r') is labeled by 1. These models are obviously disjoint since they have a different label for the edge (r, r'). Hence $\#(\Phi) = \#(\Psi_{|x}) + \#(\Psi_{|\neg x})$. □

Example 2 (Example 1 cont'd)**.** *Let us analyze the QBF Φ in Example 1. We decompose $\#(\Phi)$ into the sum of counts for $\#(\Phi_{|a})$ and $\#(\Phi_{|\neg a})$. QBF $\Phi_{|a}$ is defined as $\forall x_1 x_2 \exists y_1 y_2.(x_1 \vee y_1) \wedge (x_2 \vee y_2)$, while $\Phi_{|\neg a}$ is defined as $\forall x_1 x_2 \exists y_1 y_2.(x_2 \vee y_2)$. Branching on the variable x_2 in $\Phi_{|\neg a}$ results in $\#(\Phi_{|\neg a}) = \#((\Phi_{|\neg a})_{|x_2}) \times \#((\Phi_{|\neg a})_{|\neg x_2})$. As $\#((\Phi_{|\neg a})_{|x_2}) = \#(\forall x_1 \exists y_1 y_2.\top) = 16$ (refer to Example 5 for details) and $\#((\Phi_{|\neg a})_{|\neg x_2}) = \#(\forall x_1 \exists y_1.\top) = 4$, we obtain $\#(\Phi_{|\neg a}) = 64$. Later, in Example 3, we will demonstrate that $\#(\Phi_{|a}) = 16$. Notably, we observe that the correct tree model count is achieved, totaling $16 + (4 \times 16) = 80$.*

If $\Phi = QX.\Psi$, the order of the variables in X can be rearranged without affecting the model count for Φ. Thus, Theorem 1 holds for any variable $x \in X$. This property is utilized by the `selectVar` heuristic on Line 13 in Algorithm 1. In practice, we employ the same heuristics as `cachet` [Sang *et al.*, 2004], i.e VSADS [Sang *et al.*, 2005], striving to partition the formula's matrix into connected components for leveraging Proposition 1.

3.3 Connected Components

This optimization is a generalization exploited in many top-down model counters, such as D4 [Lagniez and Marquis, 2019]: decomposability. At its core, decomposability is a property of a CNF formula. A CNF formula $\phi(X)$ over variables X is deemed decomposable if there exists a partition $X_1, \ldots, X_k$ of X such that $\phi(X) = \phi_1(X_1) \wedge \cdots \wedge \phi_k(X_k)$. In this case, it is evident that $\#(\phi) = \prod_{i=1}^{k} \#(\phi_i)$. Extending this property, we generalize it to estimate the number of models of QBF whose matrix is decomposable:

Proposition 1. *Let Φ be a QBF of the form $\forall X_1 \exists Y_1 \ldots \forall X_k \exists Y_k . \phi_1 \wedge \phi_2$ such that $X = \biguplus_{i=1}^{k} X_i$, $Y = \biguplus_{i=1}^{k} Y_i$ and $Var(\phi_1) \cap Var(\phi_2) \subseteq X$. We define $Y_i^1 = Y_i \cap Var(\phi_1)$ and $Y_i^2 = Y_i \cap Var(\phi_2)$. Then $\#(\Phi) = \#(\Phi_1) \times \#(\Phi_2)$ where $\Phi_1 = \forall X_1 \exists Y_1^1 \ldots \forall X_k \exists Y_k^1 . \phi_1$ and $\Phi_2 = \forall X_1 \exists Y_1^2 \ldots \forall X_k \exists Y_k^2 . \phi_2$.*

Proof. The proof is by induction on the number k of blocks of universal quantifiers. First, assume that $k = 0$, that is, there is no block of universal quantifiers. In other words, $\Phi = \exists Y . \phi_1 \wedge \phi_2$. In this case, $\#(\Phi)$ is the number of models of $\phi_1 \wedge \phi_2$ which is equal by definition to the propositional model count $\#SAT(\phi_1 \wedge \phi_2)$, which, by what precedes, is equal to $\#SAT(\phi_1) \times \#SAT(\phi_2)$. But again by definition, $\#SAT(\phi_i) = \#(\Phi_i)$ for $i \in \{1, 2\}$. Hence $\#(\Phi) = \#(\Phi_1) \times \#(\Phi_2)$ which establishes the base case of the induction. Now assume $\Phi = \forall X_1 \exists Y_1 \ldots \forall X_{k+1} \exists Y_{k+1} . \phi_1 \wedge \phi_2$. By recursively applying Theorem 1, we have:

$$\#\Phi = \prod_{A_1 \in 2^{X_1}} \sum_{E_1 \in 2^{Y_1}} \#\Phi_{|A_1, E_1}.$$

Now, observe that for every $A_1 \in 2^{X_1}$ and $E_1 \in 2^{Y_1}$:

$$\begin{aligned}\Phi_{|A_1,E_1} &= \forall X_2 \exists Y_2 \ldots \forall X_{k+1} \exists Y_{k+1} . \phi_{1\ |A_1,E_1} \wedge \phi_{2\ |A_1,E_1} \\ &= \forall X_2 \exists Y_2 \ldots \forall X_{k+1} \exists Y_{k+1} . \phi_{1\ |A_1,E_1^1} \wedge \phi_{2\ |A_1,E_1^2}\end{aligned}$$

where E_1^1 (resp. E_1^2) is defined as E_1 restricted to variables in Y_1^1 (resp. Y_1^2). For a QBF Φ with fewer than k universal quantifier blocks, by induction, we have $\#(\Phi_{|A_1,E_1}) = \#(\Psi_1) \times \#(\Psi_2)$, where $\Psi_i = \forall X_2 \exists Y_2^i \ldots \forall X_{k+1} \exists Y_{k+1}^i . \phi_{i\ |A_1,E_1^i}$. Note that $\Psi_i = \Phi_i|_{A_1,E_1^i}$. Thus, $\#(\Phi_{|A_1,E_1}) = \#(\Phi_{1\ |A_1,E_1^1}) \times \#(\Phi_{2\ |A_1,Ep_1^2})$. It follows that $\#(\Phi) =$

$$\begin{aligned}&\prod_{A_1 \in 2^{X_1}} \sum_{E_1^1 \in 2^{Y_1^1}} \sum_{E_1^2 \in 2^{Y_1^2}} \#(\Phi_{|A_1,E_1}) \\ &= \prod_{A_1 \in 2^{X_1}} \sum_{E_1^1 \in 2^{Y_1^1}} \sum_{E_1^2 \in 2^{Y_1^2}} \#(\Phi_{1\ |A_1,E_1^1}) \times \#(\Phi_{2\ |A_1,E_1^2}) \\ &= \prod_{A_1 \in 2^{X_1}} \Big(\sum_{E_1^1 \in 2^{Y_1^1}} \#(\Phi_{1\ |A_1,E_1^1}) \Big) \times \Big(\sum_{E_1^2 \in 2^{Y_1^2}} \#(\Phi_{2\ |A_1,E_1^2}) \Big) \\ &= \Big(\prod_{A_1 \in 2^{X_1}} \sum_{E_1^1 \in 2^{Y_1^1}} \#(\Phi_{1\ |A_1,E_1^1}) \Big) \times \\ &\quad \Big(\prod_{A_1 \in 2^{X_1}} \sum_{E_1^2 \in 2^{Y_1^2}} \#(\Phi_{2\ |A_1,E_1^2}) \Big) = \#(\Phi_1) \times \#(\Phi_2).\end{aligned}$$

□

Example 3 (Example 1 cont'd)**.** *Let us reevaluate the formula Φ from Example 1, with a specific focus on the formula $\Phi_{|a} = \forall x_1 x_2 \exists y_1 y_2 . (x_1 \vee y_1) \wedge (x_2 \vee y_2)$ highlighted in Example 2. Notably, this formula comprises two connected components: $x_1 \vee y_1$ and $x_2 \vee y_2$. Applying Proposition 1, we determine that $\#(\Phi_{|a}) = \#(\forall x_1 x_2 \exists y_1 . (x_1 \vee y_1)) \times \#(\forall x_1 x_2 \exists y_2 . (x_2 \vee y_2)) = 4 \times 4 = 16$.*

3.4 Counting Tree Models from Cache

Caching plays a crucial role in cutting-edge model counters [Sang *et al.*, 2004; Lagniez and Marquis, 2021; van Bremen *et al.*, 2021; Bart *et al.*, 2014]. To achieve optimal efficiency, it is essential to incorporate the most pertinent information into the cache structure, thereby maximizing positive hits and minimizing the computational effort to retrieve the tree model count (the cost associated with searching in the hash table).

In propositional model counting considering the matrix alone suffices (due to all variables being existentially quantified). Determining the correct tree model count necessitates considering both the matrix and the quantifier prefix. However, using the quantifier prefix along with the matrix as the key when adding items to the cache could be suboptimal, potentially resulting in an artificial increase in negative hits.

Indeed, a naive caching strategy would involve storing the tree model count for every formula Ψ solved so far by our algorithm. A more effective strategy is employed, improving the number of cache hits by storing only the model count of a reduced QBF. To illustrate, assume a recursive call is made to count the number of models of $\Psi_1 = \forall X_0 \exists y \exists Z \forall x_1 . \phi$ for some CNF formula ϕ that does not depend on variables Z. Assume that a previous recursive call has determined that $\Psi_0 = \forall X_0 \exists y \forall x_1 . \phi$ has N models. A naive caching strategy might overlook the knowledge that $\#(\Psi_0) = N$ since $\Psi_1 \neq \Psi_0$. However, it can be observed that $\#(\Psi_1) = 2^{2^{|X_0|} \cdot |Z|} N$.

This observation can be generalized as follows: Given a prefix $\Pi = Q_1 x_1 \ldots Q_k x_k$, a *subprefix* Π' is a prefix of the form $Q_{i_1} x_{i_1} \ldots Q_{i_p} x_{i_p}$ with $i_1 < \cdots < i_p$. That is, Π' is obtained from Π by removing quantified variables but keeping the order. If Π' is a subprefix of Π and $Y \subseteq Var(\Pi) \setminus Var(\Pi')$, we denote by $\Pi' + Y$ the subprefix obtained by adding again the Y variables in Π, keeping the same order as Π. A prefix Π' is said to be a *reduction* of Π if the following hold: let y be the first existentially quantified variable of Π'. Then every universal variable x that appears after y in Π also appears in Π'. This notion is interesting because of the following lemma:

Lemma 2. *Let Π' be a reduction of Π such that $Var(\phi) \subseteq Var(\Pi')$ and let x be the last variable (for the order induced by Π) of $Var(\Pi) \setminus Var(\Pi')$. Then $\Pi'' = \Pi' + x$ is a reduction of Π and $\#(\Pi''.\phi) = (\#(\Pi'.\phi))^2$ if x is universal and $\#(\Pi''.\phi) = 2^{2^p} \#(\Pi'.\phi)$ if x is existential and if there are p universal variables before x in Π'.*

Proof. We first show that Π'' is a reduction of Π. By definition, Π'' is a subprefix of Π. Now if x is universally quantified then by definition, there is no existentially quantified variable before x in Π' since Π' is a reduction. Hence Π'' is a reduction. Now assume x is existentially quantified. Let y be the

first existentially quantified variable of Π'. If x appears after y then y is still the first existentially quantified variable of Π' and hence since Π' is a reduction, every universal variable of Π that comes after y are in Π' hence in Π'' too hence Π'' is a reduction. If x comes before y, we have to show that every universal variables of Π that are after x are in Π''. Since by definition we choose x to be the last variable of $\Pi \setminus \Pi'$, every variable of Π appearing after x are also in Π'', hence universal variables too. So Π'' is a reduction.

Now assume that x is universal. Since Π'' is a reduction, no existential variable of Π can appear before x in Π''. Hence we can apply theorem 1 on x to find that $\#(\Pi''.\phi) = (\#(\Pi'.\phi_{|x}))(\#(\Pi'.\phi_{|\neg x}))$. But $Var(\phi) \subseteq Var(\Pi')$, hence $x \notin Var(\phi)$. In other words, $\phi_{|x} = \phi_{|\neg x} = \phi$. Hence $\#(\Pi''.\phi) = (\#(\Pi'.\phi))^2$.

Finally, assume x is existential. Since $x \notin Var(\phi)$, ϕ is logically equivalent to $\phi \wedge (x \vee \neg x)$. By Proposition 1, $\#(\Pi''.\phi) = \#(\Pi'.\phi) \times \#(\forall X_1 \exists x \forall X_2.\top)$ where X_1 are the universal variables of Π'' before $\exists x$ and X_2 the universal variables after $\exists x$. An easy induction show that $\#(\forall X_1 \exists x \forall X_2.\top) = 2^{2^{|X_1|}}$ which concludes the proof. □

Applying Lemma 2 iteratively on a reduction Π' of Π until the prefix are the same leads to an efficient algorithm to compute $\#(\Pi.\phi)$ knowing $\#(\Pi'.\phi)$. The pseudocode for this approach is given in Algorithm 2. We start by collecting $Var(\Pi) \setminus Var(\Pi')$ in the for-loop line 2. We maintain a counter p which is the number of universal quantifier in Π' preceding the current examined variable. We only increase p if the quantifier is universal and if the variables appears in Π'. Each time a variable not in Π' is found, we add its quantifier to L together with the number of preceding universal quantifiers. We then go over the list L in reverse order on line 4 and apply the transformation from Lemma 2 to transform $\#(\Pi'.\phi)$ into $\#(\Pi.\phi)$. We hence have the following lemma:

Lemma 3. *Let Π be a quantifier prefix, Π' a reduction of Π and ϕ a CNF formula such that $Var(\phi) \subseteq Var(\Pi')$. Then Algorithm 2 on parameters $(\Pi, \Pi', \#(\Pi'.\phi))$ returns $\#(\Pi.\phi)$.*

Before explaining our caching strategy, we need a last notion. To maximize the number of cache hit, we would like to reduce the prefix of a QBF as much as possible, while still being able to recover the model count. We do this by definining a notion of minimal reduction of a prefix Π as follows: Given a prefix Π and a set of variables $V \subseteq Var(\Pi)$, define $\texttt{Shrink}(\Pi, V)$ to be the subprefix obtained as follows: first, remove from Π every existential variable that does not appear in V and then, if the remaining prefix starts with a universally quantified block, remove from this block (and only this one) every variable that does not appear in V. One can observe that $\texttt{Shrink}(\Pi, V)$ is a reduction of Π. Actually it is the minimal reduction Π' of Π such that $Var(\Phi) \subseteq Var(\Pi')$ in the following sense:

Lemma 4. *Let Π' be a reduction of Π such that $V \subseteq Var(\Pi')$. Then $\texttt{Shrink}(\Pi, V)$ is a reduction of Π'. In particular, $\texttt{Shrink}(\Pi, V) = \texttt{Shrink}(\Pi', V)$.*

Proof. We show that every variable x of $\texttt{Shrink}(\Pi, V)$ are also in Π'. First assume x is existentially quantified. Then by definition it is in V. Hence x is also in Π'. Now let x be a variable that is universally quantified in $\texttt{Shrink}(\Pi, V)$. Then first assume there is an existential variable y before it in $\texttt{Shrink}(\Pi, V)$. In this case, from what precedes, y is also in Π' and since Π' is a reduction, every universal variable of Π after y have to be in Π'. In particular, x is in Π'. Now, assume that only universal variables appear before x in $\texttt{Shrink}(\Pi, V)$. In this case, $x \in V$ by definition of $\texttt{Shrink}$, hence x appears in Π' since $V \subseteq Var(\Pi')$. Hence $\texttt{Shrink}(\Pi, V)$ is a reduction of Π'. We hence also that have $\texttt{Shrink}(\Pi, V)$ is a reduction of $\texttt{Shrink}(\Pi', V)$. But with the same argument, $\texttt{Shrink}(\Pi', V)$ is also a reduction of $\texttt{Shrink}(\Pi, V)$. Hence $\texttt{Shrink}(\Pi, V) = \texttt{Shrink}(\Pi', V)$. □

We are now ready to explain our caching method. We maintain a cache mapping CNF formula to an integer defined as follows: we let Π_0 be the prefix used in the very first call of Algorithm 1. Then for every CNF formulas ϕ such that $cache[\phi] \neq \emptyset$, we maintain the invariant that $cache[\phi] = \#(\Pi.\phi)$ where $\Pi = \texttt{Shrink}(\Pi_0, Var(\phi))$. We maintain also the invariant that every recursive call of Algorithm 1 on $\Pi\phi$ is such that Π is a reduction of Π_0.

Upon these invariants, line 6 of Algorithm 1 returns the correct value. Indeed, by Lemma 4, $\Pi' = \texttt{Shrink}(\Pi, Var(\phi'))$ is equal to $\texttt{Shrink}(\Pi_0, Var(\phi'))$ and by Lemma 3, $\texttt{CountReduct}(\phi', \Pi, \Pi')$ returns $\#(\Pi.\phi')$ which is equal to $\#(\Pi.\phi)$ by Lemma 1.

It is now easy to check that both invariants hold at each recursive call. Indeed, every recursive call either removes the first variable of Π' (lines 14 and 15), which transforms a reduction of Π_0 to another reduction of Π_0. Or it removes existential variables that do not appear in the formula (line 11) which also produces a reduction of Π_0 again.

Finally, when the cache is modified on line 17, the value stored for ϕ' is equal to $\#(\Pi'.\phi')$ where $\Pi' = \texttt{Shrink}(\Pi, Var(\phi'))$ by Theorem 1 and Proposition 1. By Lemma 4 $\Pi' = \texttt{Shrink}(\Pi_0, Var(\phi'))$. Hence $cache[\phi']$ is set to the desired value.

Example 4 (Example 1 cont'd)**.** *Let us revisit the formula Φ from Example 1, assuming our initial focus is on counting the $\neg a$ branch. Subsequently, $x_2 \vee y_2$ is incorporated into the cache with a count of 2. It is important to note that the relevant quantifier prefix under consideration here is $\Pi' = \forall x_2 \exists y_2$, distinct from the broader $\Pi = \forall x_1 x_2 \exists y_1 y_2$. One can observe that Π' is a reduction of Π. Hence we can iteratively apply Lemma 2 to retrieve $\#(\Pi.\phi_{|\neg a})$ from $\#(\Pi'.\phi_{|\neg a})$. The former value is hence equal to $(2^{2^1}\#(\Pi'.\phi_{|\neg a}))^2$ by applying Lemma 2 first on y_1 and then on x_1. By observing that $\#(\Pi.'\phi_{|\neg a}) = 2$, we hence deduce that $\#(\Pi.\phi_{|\neg a}) = 64$ which can be verified to be the correct model count.*

3.5 Counting Tree Models of Valid QBF Formula

When Algorithm 1 is called on a formula whose matrix is a valid CNF formula, then we can explicitly compute the number of tree models using Algorithm 2 and the right parameters, as follows:

Algorithm 2: `CountReduct`

input : $\Pi = Q_1 x_1 \dots Q_n x_n$: a quantifier prefix, Π': a reduction of Π, N: an integer
output: the number of tree models of $\Pi.\phi$ if $N = \#(\Pi'.\phi)$

1 $p \leftarrow 0$, $L \leftarrow []$;
2 **for** i **from** 1 **to** n **do**
  if $x_i \notin Var(\Pi')$ **then** $L.append((Q_i, p))$;
  else if $Q_i = \forall$ **then** $p \leftarrow p + 1$;
3 $M \leftarrow N$, $\ell \leftarrow length(L)$;
4 **for** i **from** ℓ **to** 1 **do**
  $(Q, p) \leftarrow L[i]$;
  if $Q = \forall$ **then** $M \leftarrow M^2$ **else** $M \leftarrow 2^{2^p} \cdot M$;
5 **return** M;

Corollary 1. *Let Π be a quantifier prefix. Then Algorithm 2 on parameters $(\Pi, \emptyset, 1)$ returns $\#(\Pi.\top)$.*

Proof. It is immediate from Lemma 3 since $\emptyset$ is a reduction of Π for any Π and $\#(\top) = 1$. □

Algorithm 1 uses this connection on line 4 to return the correct number of tree models when the matrix ϕ' is valid.

Example 5 (Example 1 cont'd)**.** *Let us revisit again the formula Φ in Example 1, focusing specifically on the valid formula $(\Phi_{|\neg a})_{|x_2}) = \forall x_1 \exists y_1 y_2 . \top$ obtained in Example 2. Algorithm 2 is hence called with parameters $\Pi = \forall x_1 \exists y_1 y_2$, $\Pi' = \emptyset$ and $N = 1$. The first for-loop construct the list $L = [(\forall, 0), (\exists, 0), (\exists, 0)]$. Then N is modified: the last two elements of L both induce a multiplication of N by $2^{2^0} = 2$ and the first one squares everything. Hence the algorithm returns $4^2 = 16$ achieving the correct count of tree models for the given formula.*

4 Experimental Evaluation

We implemented the algorithm introduced above in a C++ tool called d4-QBF.[1] This tool builds upon the infrastructure provided by the propositional model counter D4 [Lagniez and Marquis, 2019].[2] With our evaluation, we aim at addressing the following research questions:

RQ1. How does d4-QBF scale in terms of solved instances and runtime?
RQ2. What is the tree model count of challenging benchmarks and how can we ensure correctness of the counting algorithm?

Environment. All experiments were executed on a compute cluster with dual-socket 16 core AMD EPYC 7313 processors with 3.7 GHz and 256 GB main memory. We run all tools on a single core with a timeout of 900 seconds. The memory limit was set to 32GB.

[1] https://zenodo.org/records/11153123
[2] https://github.com/crillab/d4

	#Vars	#Clauses	#QBlocks
KBKFTrue	$8n+1$	$13n+(n-1)$	$2n+2$
KBKFTrueTseitin	$8n+1$	$18n$	$2n+2$
EQTrueNested	$3n$	$2n+1$	$2n$
EQTrue	$3n$	$2n+1$	2
ParityTrue	$2n$	$4(n-1)+2$	2

Table 1: Properties of the formula families w.r.t. parameter n.

Other tools. To the best of our knowledge, currently there exists no model counters for QBFs with arbitrary prefix. For experiments on 2QBFs (QBFs with only quantifier alternation), we run the two tools presented in [Plank *et al.*, 2023]: (1) qcounter, an enumerative solution-counter and (2) baseline, a simple baseline implementation that enumerates all assignments of the universal and calls the propositional model counter Ganak [Sharma *et al.*, 2019]. For evaluating our approach on formulas with more than two quantifier alternations, we run the two QBF solvers Caqe [Rabe and Tentrup, 2015] and DepQBF [Lonsing and Egly, 2017] which are based on orthogonal solving technology and which are top-ranked in the recent QBF competition.

Benchmarks. To evaluate our implementation, we consider seven different families of QBFs originating from two sources:

- *Unique-SAT and Random Benchmarks [Plank* et al.*, 2023].* These formulas are true 2QBFs with prefix $\forall X \exists Y$ for which the exact model count is known. The random set consists of the 3.950 true instances with $2 \leq |X|, |Y| \leq 11$. The unique-SAT set consists of 497 formulas that were build from propositional formulas with 26 to 240 variables and 61 to 643 clauses. Given a propositional formula ϕ, the negated unique-SAT QBF encoding based on [Kleine Büning *et al.*, 1995] results in a false QBF iff ϕ has exactly one solution. If the propositional formula ϕ contains n variables and has $m > 1$ models, then the resulting QBF has $(2^n)^{2^n - m}(m-1)^m$ different solutions.
- *Crafted instances [Heisinger and Seidl, 2023].* Crafted formula families play an important role in proof complexity. Because of their scalability on a parameter n they provide also expressive benchmarks for testing solvers. Usually, only false instances are considered, but [Heisinger and Seidl, 2023] presented the true formula families KBKFTrue and ParityTrue that were also part of the QBFGallery 2023.[3] In addition to these two families, we introduce three more families: KBKFTrueTseitin, EQTrue, EQTrueNested. Therefore, we extended the generator of [Heisinger and Seidl, 2023]. Properties of the formulas are shown in Table 1. Whereas ParityTrue and EqTrue are 2QBFs, KBKFTrue, KBKFTrueTseitin, and EQTrueNested have n quantifier alternations (where n is the parameter of each specific instance). For our experiments, we chose $n \in \{1, \dots, 25\}$.

[3] http://qbfeval.org

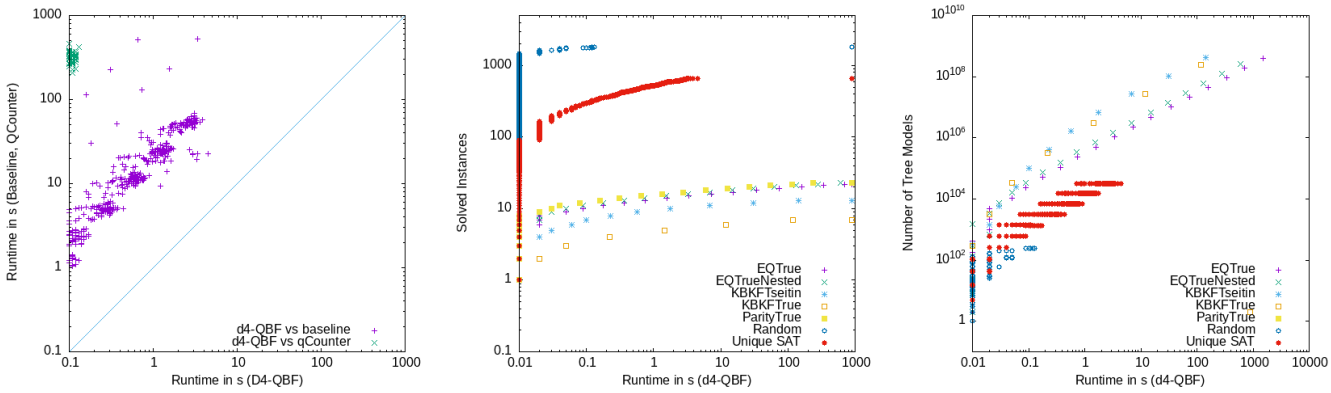

Figure 2: Runtime comparison with 2QBF counters (left). Runtimes of d4-QBF on all benchmarks (middle). Tree model count (right).

Analysis of RQ1 We compared the runtimes of d4-QBF and the 2QBF solution counters baseline and qcounter on all 2QBF benchmarks (UniqueSAT, Random, ParityTrue, and EQ). The results are shown in the left plot of Figure 2. Our tool d4-QBF solved all of the considered 2QBFs considerably faster than the two other tools. Without the optimizations presented above, our algorithm corresponds to the approach implemented in baseline. The runtimes give a clear empirical indication of the positive impact of the proposed optimizations on the solving performance.

The runtimes of d4-QBF for all considered benchmark sets are shown in the middle of Figure 2. Out of 2586 formulas, only 10 formulas could not be solved (all from the crafted formulas set). As there are currently no other solution counters for QBFs of arbitrary prefix, we run QBF solvers on the crafted formulas. The number of solved formulas are shown in Table 2. Even state-of-the-art solvers could not solve all of the benchmarks. Even more, there are some formulas that are exclusively solved by d4-QBF. These results demonstrate the practical feasibility of tree model counting even for challenging benchmarks.

Analysis of RQ2 The 2QBF benchmarks of [Plank *et al.*, 2023] are generated in such a manner that the number of tree models is known. The formulas of the ParityTrue family have only solution independent of parameter n. Because of their structured nature, we could describe the number of solutions of the other families with (non-linear) recurrences that could be confirmed by d4-QBF. As an interesting side effect, we get some novel insights on the characteristics of the crafted formula families and their solution space. So far, they have mainly been investigated from a proof complexity point of view [Beyersdorff *et al.*, 2021]. The right plot of Figure 2 shows the number of solutions for the instances solved by d4-QBF.

	DepQBF	Cage	baseline	qcounter	d4-QBF
KBKFTrue	18	19	-	-	15
KBKFTrueTseitin	18	8	-	-	24
EQTrueNested	24	24	-	-	24
EQTrue	24	24	16	6	23
ParityTrue	24	18	16	16	24

Table 2: Number of solved instances for crafted formula familes.

5 Conclusion and Perspectives

In this paper, we addressed the intricate problem of counting solutions for QBFs. Our primary contribution is the development of a comprehensive top-down tree model counter, named d4-QBF, specifically crafted to proficiently handle true QBFs with arbitrary quantifier prefix. Throughout our exploration, we provide the theory to safely incorporate a powerful branching heuristic. Further, we brought attention to the crucial considerations of connected components, free variables, and caching, all of which play key roles in ensuring the efficiency and accuracy of the solution counting process. In addition to being the first counter capable of handling arbitrary QBF formulas, our tool, d4-QBF, distinguishes itself by surpassing state-of-the-art counters specifically designed for 2QBFs. This practical superiority underscores the algorithm's robustness and positions it as a promising solution for solution counting challenges in the domain of QBFs.

Looking forward, our focus is on exploring more tailored simplification techniques to replace propositional Binary Constraint Propagation (BCP) by the QBF version QBCP as employed in [Lonsing and Egly, 2018]. Additionally, we plan to replace the initial SAT call in the algorithm with a strategy customized for QBF formulas. We aim to explore the potential effectiveness of preprocessing techniques [Wimmer *et al.*, 2019; Heule *et al.*, 2015; Cheng and Jiang, 2023], similar to their successful application in solving the #SAT problem [Lagniez and Marquis, 2017; Lagniez *et al.*, 2020]. Another key area for enhancement is the representation of the tree model count, which, at worst, necessitates an exponential number of bits. Our proposed solution involves investigating compact representations, such as polynomial representations, to achieve more efficient modeling.

Acknowledgments

This work has benefited from the support of the National Research Agency under France 2030, MAIA Project ANR-22-EXES-0009 and by project ANR KCODA, ANR-20-CE48-0004. The Institute for Symbolic Artificial Intelligence was in part supported by the LIT AI Lab funded by the State of Upper Austria.

References

[Audemard *et al.*, 2022] Gilles Audemard, Jean-Marie Lagniez, and Marie Miceli. A new exact solver for (weighted) max#sat. In *25th International Conference on Theory and Applications of Satisfiability Testing, SAT 2022*, volume 236 of *LIPIcs*, pages 28:1–28:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

[Aziz *et al.*, 2015] Rehan Abdul Aziz, Geoffrey Chu, Christian J. Muise, and Peter J. Stuckey. #∃sat: Projected model counting. In *Theory and Applications of Satisfiability Testing - SAT 2015 - 18th International Conference*, volume 9340 of *Lecture Notes in Computer Science*, pages 121–137. Springer, 2015.

[Balabanov and Jiang, 2012] Valeriy Balabanov and Jie-Hong R. Jiang. Unified QBF certification and its applications. *Formal Methods Syst. Des.*, 41(1):45–65, 2012.

[Balabanov *et al.*, 2015] Valeriy Balabanov, Jie-Hong Roland Jiang, Mikolas Janota, and Magdalena Widl. Efficient extraction of QBF (counter)models from long-distance resolution proofs. In *Proc. of the 29th AAAI Conf. on Artificial Intelligence (AAAI)*, pages 3694–3701. AAAI Press, 2015.

[Bart *et al.*, 2014] Anicet Bart, Frédéric Koriche, Jean-Marie Lagniez, and Pierre Marquis. Symmetry-driven decision diagrams for knowledge compilation. In *Proc. of 21st Europ. Conf. on Artificial Intelligence (ECAI)*, volume 263 of *Frontiers in Artificial Intelligence and Applications*, pages 51–56. IOS Press, 2014.

[Bauland *et al.*, 2005] Michael Bauland, Elmar Böhler, Nadia Creignou, Steffen Reith, Henning Schnoor, and Heribert Vollmer. Quantified constraints: The complexity of decision and counting for bounded alternation. *Electron. Colloquium Comput. Complex.*, TR05-024, 2005.

[Becker *et al.*, 2012] Bernd Becker, Rüdiger Ehlers, Matthew Lewis, and Paolo Marin. ALLQBF Solving by Computational Learning. In *Proc. of the 10th Int. Conf. on Automated Technology for Verification and Analysis (ATVA)*, volume 7561 of *LNCS*, pages 370–384. Springer, 2012.

[Beyersdorff *et al.*, 2014] Olaf Beyersdorff, Leroy Chew, and Mikolas Janota. On unification of QBF resolution-based calculi. In *Proc. of the 39th Int. Symp. on Mathematical Foundations of Computer Science (MFCS)*, volume 8635 of *LNCS*, pages 81–93. Springer, 2014.

[Beyersdorff *et al.*, 2021] Olaf Beyersdorff, Janota Mikolás, Florian Lonsing, and Martina Seidl. Quantified Boolean Formulas. In *Handbook of Satisfiability*, volume 336 of *Frontiers in Artificial Intelligence and Applications*, pages 1177–1221. IOS Press, 2021.

[Birnbaum and Lozinskii, 1999] Elazar Birnbaum and Eliezer L. Lozinskii. The good old Davis-Putnam procedure helps counting models. *J. Artif. Intell. Res.*, 10:457–477, 1999.

[Cheng and Jiang, 2023] Che Cheng and Jie-Hong R. Jiang. Lifting (D)QBF preprocessing and solving techniques to (D)SSAT. In *Proc. of the 37th AAAI Conf. on Artificial Intelligence (AAAI), 35th Conf. on Innovative Applications of Artificial Intelligence (IAAI)*, pages 3906–3914. AAAI Press, 2023.

[Chew and Slivovsky, 2022] Leroy Chew and Friedrich Slivovsky. Towards uniform certification in QBF. In *Proc. of the 39th Int. Symp. on Theoretical Aspects of Computer Science (STACS)*, volume 219 of *LIPIcs*, pages 22:1–22:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

[Dubois, 1991] Olivier Dubois. Counting the number of solutions for instances of satisfiability. *Theor. Comput. Sci.*, 81(1):49–64, 1991.

[Fremont *et al.*, 2017] Daniel J. Fremont, Markus N. Rabe, and Sanjit A. Seshia. Maximum model counting. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 3885–3892. AAAI Press, 2017.

[Giunchiglia *et al.*, 2021] Enrico Giunchiglia, Paolo Marin, and Massimo Narizzano. Reasoning with quantified boolean formulas. In *Handbook of Satisfiability - Second Edition*, volume 336 of *Frontiers in Artificial Intelligence and Applications*, pages 1157–1176. IOS Press, 2021.

[Goultiaeva *et al.*, 2011] Alexandra Goultiaeva, Allen Van Gelder, and Fahiem Bacchus. A uniform approach for generating proofs and strategies for both true and false QBF formulas. In *Proc. of the 22nd Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 546–553. IJCAI/AAAI, 2011.

[Heisinger and Seidl, 2023] Simone Heisinger and Martina Seidl. True crafted formula families for benchmarking quantified satisfiability solvers. In *Proc. of the 16th Int. Conf. on Intelligent Computer Mathematics (CICM)*, volume 14101 of *LNCS*, pages 291–296. Springer, 2023.

[Hemaspaandra and Vollmer, 1995] Lane A. Hemaspaandra and Heribert Vollmer. The satanic notations: counting classes beyond #P and other definitional adventures. *SIGACT News*, 26(1):2–13, 1995.

[Heule *et al.*, 2015] Marijn Heule, Matti Järvisalo, Florian Lonsing, Martina Seidl, and Armin Biere. Clause elimination for SAT and QSAT. *J. Artif. Intell. Res.*, 53:127–168, 2015.

[Kleine Büning and Bubeck, 2021] Hans Kleine Büning and Uwe Bubeck. Theory of quantified boolean formulas. In *Handbook of Satisfiability - Second Edition*, volume 336 of *Frontiers in Artificial Intelligence and Applications*, pages 1131–1156. IOS Press, 2021.

[Kleine Büning *et al.*, 1995] Hans Kleine Büning, Marek Karpinski, and Andreas Flögel. Resolution for quantified boolean formulas. *Inf. Comput.*, 117(1):12–18, 1995.

[Ladner, 1989] Richard E. Ladner. Polynomial space counting problems. *SIAM J. Comput.*, 18(6):1087–1097, 1989.

[Lagniez and Marquis, 2017] Jean-Marie Lagniez and Pierre Marquis. On preprocessing techniques and their impact on propositional model counting. *J. Autom. Reason.*, 58(4):413–481, 2017.

[Lagniez and Marquis, 2019] Jean-Marie Lagniez and Pierre Marquis. A recursive algorithm for projected model counting. In *Proc. of the 33rd Conf. on Artificial Intelligence (AAAI), 31st Innovative Applications of Artificial Intelligence Conf. (IAAI)*, pages 1536–1543. AAAI Press, 2019.

[Lagniez and Marquis, 2021] Jean-Marie Lagniez and Pierre Marquis. About caching in d4 2.0. In *Workshop on Counting and Sampling 2021*, 2021.

[Lagniez *et al.*, 2020] Jean-Marie Lagniez, Emmanuel Lonca, and Pierre Marquis. Definability for model counting. *Artif. Intell.*, 281:103229, 2020.

[Littman *et al.*, 1998] Michael L. Littman, Judy Goldsmith, and Martin Mundhenk. The computational complexity of probabilistic planning. *J. Artif. Intell. Res.*, 9:1–36, 1998.

[Lonsing and Egly, 2017] Florian Lonsing and Uwe Egly. Depqbf 6.0: A search-based QBF solver beyond traditional QCDCL. In *Proc. of the 26th Int. Conf. on Automated Deduction (CADE)*, volume 10395 of *LNCS*, pages 371–384. Springer, 2017.

[Lonsing and Egly, 2018] Florian Lonsing and Uwe Egly. QRAT+: generalizing QRAT by a more powerful QBF redundancy property. In *Proc. of the 9th Int. Conf. on Automated Reasoning (IJCAR)*, volume 10900 of *LNCS*, pages 161–177. Springer, 2018.

[Majercik, 2009] Stephen M. Majercik. Stochastic boolean satisfiability. In Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors, *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*, pages 887–925. IOS Press, 2009.

[Plank *et al.*, 2023] Andreas Plank, Sibylle Möhle, and Martina Seidl. Enumerative level-2 solution counting for quantified boolean formulas (short paper). In *Proc. of the 29th Int. Conf. on Principles and Practice of Constraint (CP)*, volume 280 of *LIPIcs*, pages 49:1–49:10. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023.

[Rabe and Tentrup, 2015] Markus N. Rabe and Leander Tentrup. CAQE: A certifying QBF solver. In *Proc. of the Int. Conf. on Formal Methods in Computer-Aided Design (FMCAD)*, pages 136–143. IEEE, 2015.

[Sang *et al.*, 2004] Tian Sang, Fahiem Bacchus, Paul Beame, Henry A. Kautz, and Toniann Pitassi. Combining component caching and clause learning for effective model counting. In *Proc. of the 7th Int. Conf. on Theory and Applications of Satisfiability Testing (SAT), Online Proceedings*, 2004.

[Sang *et al.*, 2005] Tian Sang, Paul Beame, and Henry A. Kautz. Heuristics for fast exact model counting. In *Proc. of the 8th Int. Conf. on Theory and Applications of Satisfiability Testing (SAT)*, volume 3569 of *LNCS*, pages 226–240. Springer, 2005.

[Sharma *et al.*, 2019] Shubham Sharma, Subhajit Roy, Mate Soos, and Kuldeep S. Meel. GANAK: A scalable probabilistic exact model counter. In Sarit Kraus, editor, *Proc. of the 28 Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 1169–1176. ijcai.org, 2019.

[Shaw *et al.*, 2024] Arijit Shaw, Brendan Juba, and Kuldeep S. Meel. An approximate skolem function counter, 2024.

[Shukla *et al.*, 2019] Ankit Shukla, Armin Biere, Luca Pulina, and Martina Seidl. A survey on applications of quantified Boolean formulas. In *Proc. of the Int. Conf. on Tools with Artificial Intelligence (ICTAI)*, pages 78–84. IEEE, 2019.

[Shukla *et al.*, 2022] Ankit Shukla, Sibylle Möhle, Manuel Kauers, and Martina Seidl. Outercount: A first-level solution-counter for quantified boolean formulas. In *Proc. of the 15th Int. Conf on Intelligent Computer Mathematics (CICM)*, volume 13467 of *LNCS*, pages 272–284. Springer, 2022.

[Stockmeyer, 1976] Larry J. Stockmeyer. The polynomial-time hierarchy. *Theor. Comput. Sci.*, 3(1):1–22, 1976.

[van Bremen *et al.*, 2021] Timothy van Bremen, Vincent Derkinderen, Shubham Sharma, Subhajit Roy, and Kuldeep S. Meel. Symmetric component caching for model counting on combinatorial instances. In *Proc. of Thirty-Fifth AAAI Conf. on Artificial Intelligence*, pages 3922–3930. AAAI Press, 2021.

[Wimmer *et al.*, 2019] Ralf Wimmer, Christoph Scholl, and Bernd Becker. The (D)QBF preprocessor HQSpre - underlying theory and its implementation. *J. Satisf. Boolean Model. Comput.*, 11(1):3–52, 2019.