Solution Counts of Some Prominent Quantified Boolean Formulas Families

Andreas Plank
Institute for Symbolic Artificial Intelligence
Johannes Kepler University Linz
Linz, Upper Austria, Austria
andreas.plank@jku.at

Manuel Kauers
Institute for Algebra
Johannes Kepler University Linz
Linz, Upper Austria, Austria
manuel.kauers@jku.at

Martina Seidl
Institute for Symbolic Artificial Intelligence
Johannes Kepler University Linz
Linz, Upper Austria, Austria
martina.seidl@jku.at

Abstract

In contrast to models of propositional formulas, which are simply Boolean variable assignments, solutions of quantified Boolean formulas (QBFs) have a tree structure reflecting the dependencies between universal and existential variables. The study of counting QBF solutions has gained momentum in recent years, but it is practically limited by the absence of benchmark sets consisting of formulas for which the number of solutions is known. In this paper, we analyse several crafted QBF formula families which are widely used in the field of proof complexity. We provide scalable benchmark sets consisting of true and false formulas that are essential for verifying the correctness of QBF solution counters.

CCS Concepts

• **Theory of computation → Logic and verification**.

Keywords

QBF, Model Counting, Formula Families

ACM Reference Format:

Andreas Plank, Manuel Kauers, and Martina Seidl. 2025. Solution Counts of Some Prominent Quantified Boolean Formulas Families. In *Proceedings of ACM SAC Conference (SAC'25)*. ACM, New York, NY, USA, Article 4, 8 pages. https://doi.org/10.1145/3672608.3707850

1 Introduction

Over the last years many problems from different application fields, like artificial intelligence, game theory and formal verification, have been solved by encoding them to quantified Boolean formulas (QBFs) [5]. Consequently, much progress has been made on solving techniques as well as on development of theory, partly based on generalizing known techniques from SAT (e.g. Q-resolution), partly being developed especially for QBFs (e.g. prenexing strategies). QBFs extend propositional formulas with existential and universal quantifiers over Boolean variables. This implies that unlike in SAT, (counter-)models of QBFs are represented in a tree structure. Each node corresponding to a universal (resp. existential) variable has two child nodes, and each node representing an existential (resp. universal) variables has one child. While the model counting problem in SAT (#SAT) is a well-established area of research, counting the number of models for QBFs (#QBF) has mostly had only theoretical contributions in the past [1, 8, 11]. The first model counters where published in recent years. In [19] an enumerative approach for counting models (i.e assignments of the existential variables such that the QBF evaluates to true) for true QBFs on the first quantifier level was introduced, utilizing blocking clauses. Dually, the authors also introduced an algorithm for counting the number of counter-models of false QBFs on the outer quantifier level. This approach was directly adopted from SAT as the solutions of variables in the outermost quantifier block are truth constants. A more general approach was presented in [17], where (counter-)model counting theory was lifted to the second quantifier level. In this setting, tree models for true QBFs and tree counter-models for false QBFs are counted. For both true and false QBFs, counting the solutions now requires capturing the dependencies of the variables to the previous quantifier levels, where blocking clauses for true QBFs are now replaced by sets of blocking Skolem functions, respectively cubes for false QBFs are replaced by sets of blocking Herbrand functions. The authors also introduced two different notions of solutions of a QBF, capturing the combinatorial possibilities based on the tree (counter-)models and the number of Skolem (resp. Herbrand) functions. An approximate model counter utilizing connections between function counting and propositional model counting was presented in [18]. The approach described in this paper showed promising results handling similar benchmark sets as state-of-the-art QBF solvers, proven to be accurate up to some theoretical assumptions.

To validate the accuracy of current and future (counter-)model counting methods, it is crucial to have diverse and scalable formulas whose number of (counter-)models is known. In this work (counter-)model counts for several widely used QBF families are presented. Furthermore, we present key metrics and experimental results from the exact QBF model counters, d4 [12] for true formulas and QCounter [17] for false formulas. Our goal is to provide insights into the efficiency of these counters, while also emphasizing the significance of encodings and the quantifier structures of the individual formulas.

First, necessary preliminaries are presented in Section 2. Then we present results for QParity formula family in Section 3, before we discuss Equality formulas in Section 4. Results for Equality formulas

This research was funded in part by the Austrian Science Fund (FWF) 10.55776/COE12 and the State of Upper Austria (LIT AI Lab). Manuel Kauers was supported by the Austrian FWF grants 10.55776/PAT8258123, 10.55776/I6130, and 10.55776/PAT9952223.

SAC'25, March 31-April 4, 2025, Sicily, Italy

ACM ISBN 979-8-4007-0629-5/25/03
https://doi.org/10.1145/3672608.3707850

with a nested quantifier structure are presented in Section 5. Finally, we present metrics and counts for formulas of the Kleine Büning, Karpinski and Flögel class, before concluding the paper in Section 7.

2 Preliminaries

We consider *quantified Boolean formulas* of the form $\Pi.\phi$. Here, $\Pi = Q_1 x_1 \dots Q_n x_n$ is called the *quantifier prefix*, where $x_1, \dots, x_n$ are pairwise distinct Boolean variables and $Q_i \in \{\forall, \exists\}$. The *matrix* ϕ is a propositional formula built from the Boolean variables of the prefix and standard logical connectives $\neg$ (negation), $\wedge$ (conjunction), $\vee$ (disjunction), $\rightarrow$ (implication), $\leftrightarrow$ (equivalence) and $\oplus$ (exclusive or). The matrix ϕ is in *conjunctive normal form (CNF)* if ϕ is a conjunction of clauses, where a clause is a disjunction of literals. A literal is a variable or a negated variable. For a literal l, $var(l) = x$ if $l = x$ or $l = \neg x$. Furthermore $\bar{l} = x$, if $l = \neg x$ and $\bar{l} = \neg x$ otherwise. For a QBF $\Phi = Q_1 x_1 \dots Q_n x_n.\phi$, the set of its variables is denoted by $var(\Phi) = \{x_1, \dots, x_n\}$. An *assignment* σ of a QBF $\Phi = \Pi.\phi$ is defined by a set of literals over (a subset of) $var(\Phi)$ such that there is no $l \in \sigma$ with $\neg l \in \sigma$. We define $var(\sigma) = \{var(l) \mid l \in \sigma\}$. If $var(\sigma) = var(\Phi)$, then σ is called a *full assignment*, otherwise it is called a *partial assignment*.

The QBF $\Phi_{|\sigma} = \Pi'.\phi|\sigma$ denotes the QBF obtained by setting all variables $x \in var(\sigma)$ to true if $x \in \sigma$ and to false if $\neg x \in \sigma$, simplifying the matrix, and removing them from Π resulting in Π'. For assigning a single variable to true (resp. to false) we also write $\Phi_{|x}$ (resp. $\Phi_{|\neg x}$). A QBF $\forall x \Pi.\phi$ is true iff $\Pi.\phi_{|x}$ and $\Pi.\phi_{|\neg x}$ are true. Dually, $\exists x \Pi.\phi$ is true iff $\Pi.\phi_{|x}$ or $\Pi.\phi_{|\neg x}$ is true. The semantics of a QBF $\Pi.\phi$ induces a variable ordering $x_i <_\Pi x_j$ for $Q_i \neq Q_j$ and $i < j$. A model of a true QBF $\Phi = \Pi.\phi$ with $|var(\Phi)| = n$ is a tree of height $n+1$ such that every node at level $k \in \{1, \dots, n\}$ in the tree corresponds to variable x_k in the ordered prefix Π. A node at level k has two children if $Q_k = \forall$ and one child if $Q_k = \exists$. A path from the root to the leaves represents a full assignment of the variables $var(\phi)$, under which the matrix ϕ evaluates to true. Dually, a counter model for a false QBF is defined as a *tree counter model* where each node at level k has two children if $Q_k = \exists$, and one child if $Q_k = \forall$. An example of a tree model is shown in Figure 2.

The number of distinct QBF (counter-) models of a QBF $\Phi = \Pi.\phi$ is denoted by $\#(\Phi)$. Notably, the number of (counter-)models is dependent on the given quantifier structure, which becomes apparent in the following theorem.

Theorem 2.1. *Let* $\Phi = Qx\Pi.\phi$. *Then* $\#(\Phi) = \#(\Pi.\phi_{|x}) \cdot \#(\Pi.\phi_{|\neg x})$ *if* $Q = \forall$ *and* $\#(\Phi) = \#(\Pi.\phi_{|x}) + \#(\Pi.\phi_{|\neg x})$ *if* $Q = \exists$.

Proof. First, assume that $Q = \forall$. Models of Φ are the set of trees whose root corresponds to the variable x and is connected to two sub-trees T_0 and T_1 such that T_0 is a model of $\Phi_{|\neg x}$ and T_1 is a model of $\Phi_{|x}$. There are $\#(\phi_{|x}) \cdot \#(\phi_{|\neg x})$ such trees hence $\#(\Phi) = \#(\phi_{|x}) \cdot \#(\phi_{|\neg x})$.

Now assume $Q = \exists$. Again, the models of Φ are the set of trees whose root r is labeled by x. If r is connected to a subtree T with root r' via an edge (r, r') that is labeled with a 0 (i.e. variable x is set to false), then T is a model of $\phi_{|\neg x}$. If the edge (r, r') is labeled with a 1 (i.e. variable x is set to true), then T is a model of $\phi_{|x}$. This implies that there are $\#(\phi_{|\neg x})$ models of Φ such that (r, r') is labeled by 0 and $\#(\phi_{|x})$ models of Φ such that (r, r') is labeled by 1. Since different labels of the edges (r, r') imply disjoint models of the QBF, the model count can be described by $\#(\Phi) = \#(\Pi.\phi_{|x}) + \#(\Pi.\phi_{|\neg x})$. □

$$
\begin{aligned}
&\Phi_k = \forall x_1 \dots x_n \ \exists y \ \exists z_2 \dots z_n \\
&(\bar{z}_2 \vee x_1 \vee x_2) \quad \wedge \quad (\bar{z}_2 \vee \bar{x}_1 \vee \bar{x}_2) \\
&(z_2 \vee \bar{x}_1 \vee x_2) \quad \wedge \quad (z_2 \vee x_1 \vee \bar{x}_2) \\
&(\bar{z}_i \vee z_{i-1} \vee x_i) \quad \wedge \quad (\bar{z}_i \vee \bar{z}_{i-1} \vee \bar{x}_i) \quad \text{for } 3 \leq i < k \\
&(z_i \vee \bar{z}_{i-1} \vee x_i) \quad \wedge \quad (z_i \vee z_{i-1} \vee \bar{x}_i) \quad \text{for } 3 \leq i < k \\
&(y \vee z_n) \quad \wedge \quad (\bar{y} \vee \bar{z}_n)
\end{aligned}
$$

Figure 1: Structure of formulas of the Parity True formula for $k > 1$.

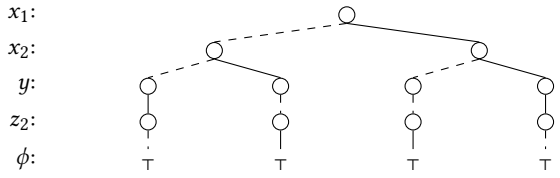

Figure 2: Model of the PARITYTrue Formula with $n = 2$, where dashed lines indicate that the according variable is set to false. Solid lines indicate that the variable is set to true.

3 The QParity Formula Family

Parity formulas [4] are the classic example of formulas not efficiently computable in AC^0 [6], a certain class in circuit complexity. In proof complexity, this hardness is employed to argue that formulas of the Parity family do not have short proofs in Q-resolution. As a consequence, several separations to other proof systems that have short proofs for these formulas follow. We will discuss both true and false parity formulas that share a similar structure except for the prefix.

3.1 The ParityTrue Family

Description of the Formula Family: Parity formulas are of the form $PARITYTrue(x_1, ..., x_n) = (x_1 \oplus \dots \oplus x_n)$ (where $\oplus$ denotes the exclusive-or), and are used to determine if the number of true variables is odd [4]. Parity formulas are the classic example of formulas that can not be represented by AC^0 [6] circuits.

QBFs based on the parity formula family are of the form $PARITYTrue(x_1, \dots, x_n, y) = ((x_1 \oplus \dots \oplus x_n) \oplus y)$, where the prefix for true formulas is $\forall x_1 \dots x_n \exists y$ [7]. Parity formulas can be classified by the parameter n, quantifying the number of universally quantified variables and clauses.

Transformation to PCNF is done using Tseitin transformation [20], where the definition variables z_j are appended to the innermost quantifier block. The resulting formula in PCNF has prefix $\forall x_1 \dots x_n \ \exists y \ \exists z_2 \dots z_n$ and clauses (with $2 < i \leq n$) as depicted in Figure 1.

Solution Count: When analyzing the given formula it is sufficient to consider the original formula in non-CNF. All Tseitin

variables $z_2 \dots z_n$ are bound by the values of the universal variables $\forall x_1, \dots, x_n$, and therefore can not add additional models to the model count.

Theorem 3.1. *For $n \in \mathbb{N}, n > 2$ the model count for the n^{th} true parity formula is*

$$\#(PARITYTrue(n)) = 1.$$

Proof. In order to satisfy each sub-tree generated by one complete assignment of the universal variables, y has to be set to the same value as $\neg((x_1 \oplus \dots \oplus x_n))$ under the respective assignment of the universal variables. This means that for each subtree there is only one possible choice for the existential variable y. The values of the other existential variables are fixed by the formula they define. In consequence, the total model count is 1. □

3.2 The Parity False Family

The structure of Parity False formulas is identical to that of Parity True formulas; however, the only difference is that some quantifiers in the prefix are switched. Hence, formulas with generation parameter n have prefix $\exists x_1 \dots x_n \forall y \exists z_2 \dots z_n$ and clauses $(2 < i \leq n)$ as depicted in Figure 1.

Solution Count: Contrary to true parity formulas, we can no longer satisfy the equation $\neg((x_1 \oplus \dots \oplus x_n) \oplus y)$, as y can always be chosen to the opposite polarity of $(x_1 \oplus \dots \oplus x_n)$. However, this is the only counter-model of the formula, hence we can formulate the following theorem:

Theorem 3.2. *For $n \in \mathbb{N}, n > 2$ the counter-model count for the nth false parity formula is*

$$\#(PARITYFalse(n)) = 1.$$

Proof. Dual to the proof of Theorem 3.1. □

3.3 Evaluation

Key metrics for true and false parity formulas are summarized in Table 1, where we also included runtime results of the model counter d4 [12], which has been modified to support QBF formulas for true formulas, and QCounter [16] for false formulas. All experiments in this paper were performed on a cluster of dual-socket AMD EPYC 7313 @ 16 × 3.7GHz machines with 32GB memory limit and 1800 seconds as timeout (TO).

Noticeably, we observe better runtime results for d4 across all input parameters, which is in accordance with the results presented in [12]. Although the number of solutions is constantly one, the runtimes of the counters reflect the hardness of the problem in their runtime. For QCounter, the formulas become hard already at a very small k. This is not surprising, because internally it relies on the solver DepQBF [13] which is a solver based on Q-resolution [9] for which the parity formulas are inherently hard.

4 Equality Formulas

Equality formulas were first introduced in [3] in the context of game theory, defining formulas with unique winning strategies

	# Variables and Clauses				Time (T)		Time (F)
n	#V	#Cls	#∃V	#∀V	d4	QC	QC
2	4	6	2	2	0.02	0.13	0.14
5	10	18	5	5	0.01	0.14	0.14
10	20	38	10	10	0.02	1.35	0.58
15	30	58	15	15	0.16	TO	915.12
20	40	78	20	20	5.89	TO	TO
25	50	98	25	25	236.86	TO	TO
k	$2k$	$4k-2$	k	k	-	-	-

Table 1: Key metrics and runtime for true and false parity formulas. The results produced by the model counter d4 reflect the true variant of the parity formulas. In contrast, the results from the (counter-)model QCounter (QC) describe outcomes for both true and false parity formulas.

for each instance. These formulas are proven to be hard for QU-resolution [21], a proof system that is more powerful than Q-resolution as shown in [3].

As for Parity Formulas, both true and false variants can be defined. The differences lie in the quantifier prefix, while the propositional matrix remains identical for both variants.

4.1 The EqualityTrue Family

Originally, the Equality formulas are a family of false formulas [3]. Following the idea of [7] these formulas can be converted into true formulas by modifying the prefix. The formulas describe the equality of n existential and n universal variables, where at least one of the equalities needs to be satisfied (see next section). For the nth equality formula this property can be expressed with the QBF $\Pi.\phi$ where

$$\Pi = \forall x_1, \dots, x_n \exists y_1, \dots, y_n \exists l_1 \dots l_n$$

and

$$\phi = \bigvee_{i=1}^{n} (x_i \leftrightarrow y_i)$$
$$\equiv (l_1 \rightarrow (x_1 \leftrightarrow y_1)) \wedge \dots \wedge (l_n \rightarrow (x_n \leftrightarrow y_n)) \wedge (l_1 \vee \dots \vee l_n).$$

Hereby Plaisted and Greenbaum [14] transformation was used to minimize the number of generated clauses in the resulting CNF. Finally, each sub-formula $(l_i \rightarrow (x_i \leftrightarrow y_i))$ can be converted to CNF by rewriting it to $(\neg l_i \vee x_i \vee \neg y_i) \wedge (\neg l_i \vee \neg x_i \vee y_i)$. The resulting formula for $k > 1$ is depicted in Figure 3.

Solution Count: Since $V = \{x_1, \dots, x_n\}$ are universal variables, we can fix an assignment over V and use Theorem 2.1 to compute

$$\begin{aligned}
&\Phi_k = \forall x_1 \dots x_k \exists y_1 \dots y_k \exists l_1 \dots l_k \\
&(\neg x_i \vee y_i \vee \neg l_i) \quad \wedge \quad \text{for } 1 \leq i \leq k \\
&(x_i \vee \neg y_i \vee \neg l_i) \quad \wedge \quad \text{for } 1 \leq i \leq k \\
&(l_1 \vee \dots \vee l_k)
\end{aligned}$$

Figure 3: Structure of formulas of the Equality True formula for $k > 1$.

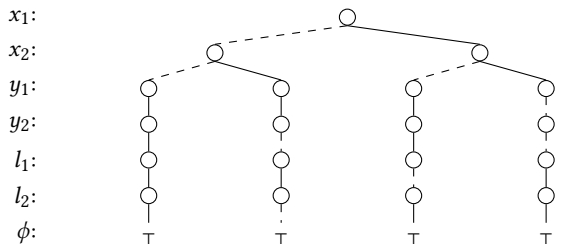

Figure 4: Model of the EqualityTrue Formula with $n = 2$, where dashed lines indicate that the according variable was set to false, else the variable was set to true.

the model count by multiplying the propositional model counts of the resulting formulas: $\#\Phi = \prod_{\sigma \in 2^V} \#\phi_{|\sigma}$ where 2^V denotes the sets of all possible assignments of variables V. It is easy to show that $\#\phi_{|\sigma} = \#\phi_{|\tau} = m$ for any $\sigma, \tau \in 2^V$, hence $\#\Phi = m^{2^n}$.

Now, let x_i^k (resp. y_i^k), be the ith, $1 \leq i \leq k$, out of k chosen variables of the set $\{x_1, \ldots, x_n\}$ (resp. set $\{y_1, \ldots, y_n\}$). The number of different possibilities to choose k universal variables $\{x_1^k, \ldots, x_k^k\}$ and existential variables $\{y_1^k, \ldots, y_k^k\}$ such that $x_i^k = y_i^k$ for all $i \in \{1, \ldots, k\}$ is $\binom{n}{k}$. In order to satisfy a QBF encoding the equality problem we still need to set values for the variables $\{l_1, \ldots, l_n\}$. Excluding the assignment which sets all $\{l_1, \ldots, l_n\}$ which always falsifies the last clause of the formula, we can set k or these variables freely. This is easy to see as k clauses are already satisfied by the satisfied k equalities. Consequently, the number of possible assignments of the $\{l_1, \ldots, l_n\}$ is given by $2^k - 1$.

For a single assignment of $\{x_1, \ldots, x_n\}$ the number of possible models for having exactly k satisfying assignments is given by $\binom{n}{k}(2^k - 1)$. Summing over all values k and considering all possible assignments $\{x_1, \ldots, x_n\}$ as calculated above yields

$$\left(\sum_{k=1}^{n} \binom{n}{k} (2^k - 1) \right)^{2^n}. \tag{1}$$

A closed form of Equation (1) is stated in the following theorem:

Theorem 4.1. *For $n \in \mathbb{N}, n > 2$ the model count for the n^{th} EqualityTrue formula is*

$$\#(EQTrue(n)) = (3^n - 2^n)^{2^n}. \tag{2}$$

Proof. We have $\sum_{k=1}^{n} \binom{n}{k}(2^k - 1) \overset{2^0=1}{=} \sum_{k=0}^{n} \binom{n}{k}(2^k - 1) = \sum_{k=0}^{n} \binom{n}{k} 2^k - \sum_{k=0}^{n} \binom{n}{k} = 3^n - 2^n$, using the binomial theorem $\sum_{k=0}^{n} \binom{n}{k} a^k = (a+1)^n$ in the last step.

□

4.2 The EqualityFalse Family

Similar to before, formulas from the EqualityTrue formula family [3] describe the equality of n existential and n universal variables. However, by flipping the quantifiers of the first two blocks, we now seek n existential variables such that at least one is equal to n universal variables. Since it is always possible to set each variable to have the opposite value of the corresponding variable, which results in a false formula.

$$\begin{aligned} &\Phi_k = \exists x_1 \ldots x_k \forall y_1 \ldots y_k \exists l_1 \ldots l_k \\ &(\neg x_i \vee y_i \vee \neg l_i) \quad \wedge \quad \text{for } 1 \leq i \leq k \\ &(x_i \vee \neg y_i \vee \neg l_i) \quad \wedge \quad \text{for } 1 \leq i \leq k \\ &(l_1 \vee \cdots \vee l_k) \end{aligned}$$

Figure 5: Structure of formulas of the EqualityFalse formula for $k > 1$.

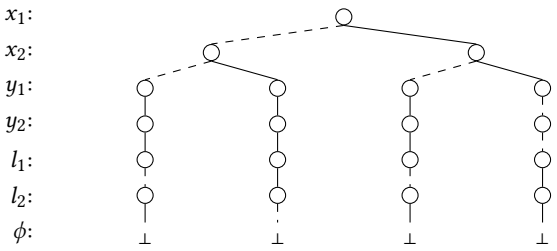

Figure 6: Model of the EqualityFalse Formula with $n = 2$, where dashed lines indicate that the according variable was set to false, else the variable was set to true.

For the nth EqualityFalse formula this property can be expressed with the QBF $\Pi.\phi$ where

$$\Pi = \exists x_1 \ldots x_n \forall y_1 \ldots y_n \exists l_1 \ldots l_n$$

and

$$\phi = (l_1 \rightarrow (x_1 \leftrightarrow y_1)) \wedge \cdots \wedge (l_n \rightarrow (x_n \leftrightarrow y_n)) \wedge (l_1 \vee \cdots \vee l_n).$$

Each sub-formula $(l_i \rightarrow (x_i \leftrightarrow y_i))$ can be converted to CNF by rewriting it to $(\neg l_i \vee x_i \vee \neg y_i) \wedge (\neg l_i \vee \neg x_i \vee y_i)$. The resulting formula for $k > 1$ is depicted in Figure 5.

Solution Count: As mentioned above, for an assignment to falsify an EqualityFalse formula, all universal variables must be set to the opposite value of their corresponding variables (i.e. $x_i \nleftrightarrow y_i$). Hence, this assignment is the only possible counter-model of a formula of the EqualityFalse family.

Theorem 4.2. *For $n \in \mathbb{N}, n > 2$ the counter-model count for the nth EqualityFalse formula is*

$$\#(EQFalse(n)) = 1. \tag{3}$$

4.3 Evaluation

Key metrics for the Equality formula family are summarized in Table 2, including the runtime results of the model counter d4 for the true variant, and QCounter for the false variant. Notably, d4 consistently outperforms QCounter across all input parameters n, which is particularly remarkable given the exponential growth in models for the EqualityTrue formulas.

5 The EqualityTrueNested Family

Formulas for the EqualityTrueNested family implement the same behavior as the preceding formulas EqualityTrue of Section 4.1. The propositional matrices are identical, however contrary to the

	# Variables and Clauses				#Solutions		Runtime (in s)	
n	#Variables	#Clauses	#∃Variables	#∀Variables	True	False	d4	QCounter
2	6	5	4	2	625	1	0.01	0.15
5	15	11	10	5	$2.38 \cdot 10^{74}$	1	0.02	0.13
10	30	21	20	10	$8.75 \cdot 10^{4877}$	1	0.06	10.51
15	45	31	30	15	$1.27 \cdot 10^{234482}$	1	0.78	TO
20	60	41	40	20	$9.25 \cdot 10^{10005820}$	1	34.28	TO
25	75	51	50	25	$2.08 \cdot 10^{400237740}$	1	1525.50	TO
k	$2 \cdot k$	$4 \cdot k - 2$	k	k	$(3^k - 2^k)^{2^k}$	1	-	-

Table 2: Key metrics and runtime for Equality formulas. The results produced by the model counter d4 reflect the true variant of the Equality formulas. In contrast, the results from the (counter-)model counter QCounter describe the corresponding outcomes for false Equality formulas.

aforementioned EqualityTrue formulas, the quantifiers of the prefix are now nested. Even though this does not influence the inherent meaning of the formula, Theorem 2.1 indicates that the model count differs from the non-nested case. Another factor that we have to consider is the encoding used to transform the formula into CNF.

5.1 Tseitin Encoding

The structure of the formula using a Tseitin encoding can be described as follows:

$$\Pi = \forall x_1 \exists y_1 \dots \forall x_n \exists y_n \exists l_1 \dots l_n$$

and

$$\phi = (l_1 \leftrightarrow (x_1 \leftrightarrow y_1)) \wedge \dots \wedge (l_n \leftrightarrow (x_n \leftrightarrow y_n)) \wedge (l_1 \vee \dots \vee l_n). \quad (4)$$

Each sub-formula $(l_i \rightarrow (x_i \leftrightarrow y_i))$ can be converted to CNF by rewriting it to $(\neg l_i \vee x_i \vee \neg y_i) \wedge (\neg l_i \vee \neg x_i \vee y_i) \wedge (l_i \vee \neg x_i \vee y_i) \wedge (l_i \vee x_i \vee \neg y_i)$. The resulting propositional matrix is equal to the matrix shown in Figure 5.

Solution Count: Analyzing the propositional matrix of the formula using Tseitin transformation, it is apparent that the Tseitin variables $l_1, \dots, l_n$ are fully determined by the values of the corresponding equivalences $(x_1 \leftrightarrow y_1)$ to $(x_n \leftrightarrow y_n)$. Hence we can omit these variables from our calculations without changing the model count of the underlying formula. The model count of the formula can then be defined using a case distinction on the first equivalence $(x_1 \leftrightarrow y_1)$. If this equivalence evaluates to false, one of the other $n-1$ equivalences has to hold in order to satisfy the formula. This implies that the model count in this case is equal to $\#EQTrueNested(n-1)$. In the other case where the first equivalence $(x_1 \leftrightarrow y_1)$ holds, equation (4) evaluates to true regardless of the values of all other *equivalences*. Hence, in this case we have to consider all combinations of the values the variables $x_2, \dots, x_n, y_2, \dots, y_n$, which are aggregated according to Theorem 2.1, where the products can be replaced by powers of two due to the symmetry of the formula. Consequently the number of solutions $f(n)$ in this case satisfies the recursion $f(n) = (f(n-1) \cdot 2)^2, n \geq 2$, and $f(1) = 1$. Finally when we add the count of both cases (y_1 is an existential variable), we square the result (x_1 is an universal variable) to obtain the final result.

Theorem 5.1. *For $n \in \mathbb{N}, n > 2$ the model count for the nth EqualityTrueNested formula using Tseitin transformation is*

$$\#(EQTrueNested(n)) = (\#(EQTrueNested(n-1)) + 2^{2^n-2})^2 \quad (5)$$

$$\#(EQTrueNested(1)) = 1 \quad (6)$$

Proof. It remains to show that the recursion $f(n) = (f(n-1) \cdot 2)^2, n \geq 2$, with $f(1) = 1$ has the solution 2^{2^n-2} for $n \geq 2$. $f(2) = (1 \cdot 2)^2 = 4$ which is equal to $2^{2^2-2} = 4$, hence the initial case is fulfilled. Now assume the relation holds for n, and we show it for $n+1$. Since $f(n+1) = (f(n) \cdot 2)^2 = (2^{2^n-2} \cdot 2)^2 = 2^{2^{n+1}-2}$ we have proven that $f(n) = 2^{2^n-2}$ for all n. Hence equation (5) follows. □

5.2 Plaisted and Greenbaum Encoding

The formula obtained from the Plaisted and Greenbaum encoding closely resembles the formula derived from the Tseitin encoding. Instead of the equivalences used before, now implications are used, increasing the number of models we can find.

The structure of the formula using a Plaisted and Greenbaum encoding is as follows:

$$\Pi = \forall x_1 \exists y_1 \dots \forall x_n \exists y_n \exists l_1 \dots l_n$$

and

$$\phi = (l_1 \rightarrow (x_1 \leftrightarrow y_1)) \wedge \dots \wedge (l_n \rightarrow (x_n \leftrightarrow y_n)) \wedge (l_1 \vee \dots \vee l_n).$$

Each sub-formula $(l_i \rightarrow (x_i \leftrightarrow y_i))$ can be converted to CNF by rewriting it to $(\neg l_i \vee x_i \vee \neg y_i) \wedge (\neg l_i \vee \neg x_i \vee y_i)$.

Solution Count: Computing the number of models is now a bit more sophisticated since the values for $l_1, \dots, l_n$, now are not fully dependent on the corresponding values of $(x_k \leftrightarrow y_k), k \in \{1, \dots, n\}$, but can be chosen freely under some constraints. More specifically, when $(x_k \leftrightarrow y_k)$ evaluates to false for a specific index k, the corresponding variable l_k can be set arbitrarily as long as at least one value $l_j, j \in \{1, \dots, n\}$ is true. This behavior can be described using function $a(k, l)$, which takes the number of variables l which can be chosen arbitrarily, as well as the level in the assignment-tree k (which again influences the aggregation we have to perform according to Theorem 2.1). The base case at the lowest level of the assignment-tree is defined by $a(0, l) = 2^l - 1$, which describes

the combinations of the variables l which can be chosen arbitrarily minus the assignment where all $l_k, k \in \{1, \ldots, n\}$ are set to false, which would falsify the last clause of the formula. For level $k \in \{2, \ldots, n\}$ the recursion can be defined according to Theorem 2.1, which yields $a(k, l) = (a(k-1, l+1)+(a(k-1, l))^2$. The number of models for generation parameter n can then be simply read of by setting the $k = n$ and $l = 0$ (i.e. at level n of the assignment-tree there are no longer any other possibilities for parameter l). This computation is also depicted in Figure 7 and can be quantified by the following theorem.

Theorem 5.2. *For $n \in \mathbb{N}, n > 2$ the model count for the nth EqualityTrueNested formula is*

$$\#(EQTrueNested(n)) = a(n, 0) \tag{7}$$

where

$$a(k, l) = (a(k-1, l+1) + a(k-1, l))^2 \tag{8}$$

$$a(0, l) = 2^l - 1 \tag{9}$$

Proof. The straightforward induction proof of the closed form of $a(k, l)$ is omitted for brevity. □

5.3 Plaisted and Greenbaum Encoding – Alternative Quantifier Structure

Another method to modify the formula involves shifting the quantifiers of the variables $l_k, k \in \{1, \ldots, n\}$ in the prefix, to the corresponding quantifier blocks of x_k and y_k. As discussed above, Tseitin variables do not affect the number of models for this formula family. Hence, we do not need to calculate the model count anew for this encoding. The formula with a Plaisted and Greenbaum encoding looks as follows.

$$\Pi = \forall x_1 \exists y_1 \exists l_1 \cdots \forall x_n \exists y_n \exists l_n$$

and

$$\phi = (l_1 \rightarrow (x_1 \leftrightarrow y_1)) \wedge \cdots \wedge (l_n \rightarrow (x_n \leftrightarrow y_n)) \wedge (l_1 \vee \cdots \vee l_n).$$

As before each sub-formula $(l_i \rightarrow (x_i \leftrightarrow y_i))$ can be converted to CNF by rewriting it to $(\neg l_i \vee x_i \vee \neg y_i) \wedge (\neg l_i \vee \neg x_i \vee y_i)$.

Solution Count: The model count of this formula family can be computed recursively using Theorem 3.1. Notably, the formula is completely symmetric with respect to the universal quantifiers, which allows us to replace the products of Theorem 3.1 for the universal variable on the outer quantifier level with powers of two. Further – without loss of generality – we only need to consider the case where $x_1 = \top$, and square the count for the sub-tree. We can further distinguish cases based on the value of the existential variable y_1. When $y_1 = \bot$ (i.e. $x_1 \nleftrightarrow y_1$), the relation $(l_1 \rightarrow (x_1 \leftrightarrow y_1))$ requires the Tseitin variable l_1 to be set to $\bot$. This means that the model counting problem for the rest of the formula simplifies to counting the number of models for $EQTrueNested(n-1)$. By applying similar reasoning, we can conclude that the formula reduces to the problem of counting the number of models for $EQTrueNested(n-1)$ under the assignments $y_1 = \top$ and $l_1 = \bot$. If $y_1 = \top$ and $l_1 = \top$, the clause $(l_1 \vee \cdots \vee l_k)$ is true, hence all other variables can be set arbitrarily as long as $(l_i \rightarrow (x_i \leftrightarrow y_i))$ is true for all $i \in \{2, \ldots, n\}$. By applying Theorem 2.1, we obtain 3^{2^n-2} such assignments. Since y_1 and l_1 are existential variables, we have to sum these counts, square the result due to the universal variable x_1, and obtain the total model count for the formula.

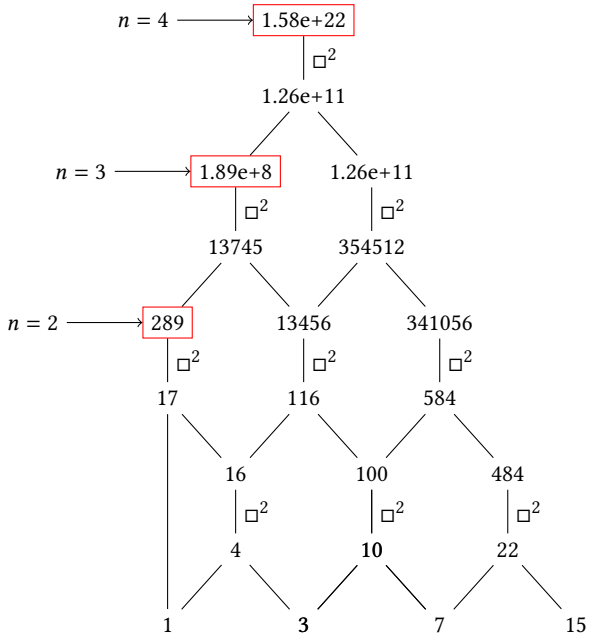

Figure 7: Computation tree of the number of models up to $n = 4$ for True Nested Equality formulas

Theorem 5.3. *For $n \in \mathbb{N}, n > 2$ the model count for the nth EqualityTrueNested formula is*

$$\begin{aligned} &\#(EQTrueNested(n)) \\ &= (2 \cdot \#(EQTrueNested(n-1)) + 3^{2^{n-1}-2})^2 \end{aligned} \tag{10}$$

5.4 Evaluation

Similar to before, we discuss key metrics for EqualityTrue formulas with nested quantifiers and highlight the differences in runtime for the model counter d4. Interestingly, despite the fact that the structure of the formulas, as well as the number of clauses and variables are equal to the EqualityTrue formulas, we see noticeably different runtime results for the model counter d4. The evaluation results for the quantifier structure described in Section 5.1 to Section 5.3 are summarized in Table 3.

6 Kleine Büning, Karpinski and Flögel Formulas (KBKF Family)

The formulas introduced by Kleine Büning, Karpinski and Flögel, commonly referred to as formulas of the KBKF family [10], were the first formulas are proven to be hard for Q-resolution. These formulas play a central role in QBF proof complexity theory, analogous to the pigeonhole problem for propositional logic [2]. Originally, the formulas of the KBKF formula family are false, but true variants have been introduced as well [7]. Furthermore, there exist differentiate variants of Q-resolution, resulting in a slightly different formula structures as described in [4]. However, we focus on formulas based

	# Variables and Clauses				#Solutions		Runtime - d4 (in s)	
n	#Vars	#Clauses	#∃Vars	#∀Vars	Classical	Alternate	Classical	Alternate
2	6	5	4	2	289	121	0.01	0.01
5	15	11	10	5	$6.53 \cdot 10^{54}$	$8.69 \cdot 10^{28}$	0.01	-
10	30	21	20	10	$5.02 \cdot 10^{3318}$	$3.71 \cdot 10^{120}$	0.02	-
15	45	31	30	15	$1.24 \cdot 10^{155542}$	$1.15 \cdot 10^{31268}$	0.37	-
20	60	41	40	20	$9.87 \cdot 10^{6555633}$	MO	14.76	-
25	75	51	50	25	$2.04 \cdot 10^{260284763}$	MO	605.65	-
k	$3 \cdot k$	$2 \cdot k + 1$	$2 \cdot k$	k	Eq. (7)	Eq. (10)	-	-

Table 3: Key metrics and runtime for Nested Equality formulas with classical and alternate quantifier structure.

$$\Phi_1 = \exists d_1 e_1 \forall x_1 \exists f_1 . (\neg d_1 \vee \neg e_1) \wedge (d_1 \vee x_1 \vee \neg f_1) \wedge (e_1 \vee \neg x_1 \vee \neg f_1) \wedge (x_1 \vee f_1) \wedge (\neg x_1 \vee f_1)$$

(a) Structure of formulas of the KBKF formula family based on Q-resolution for $k = 1$

$$\begin{aligned}
\Phi_k = \exists d_1, e_1 \forall x_1 \ldots \exists d_k e_k \forall x_k \exists f_1 \ldots f_k . \\
(\neg d_1 \vee \neg e_1) \quad & \wedge \\
(d_i \vee x_i \vee \neg d_{i+1} \vee \neg e_{i+1}) \quad & \wedge \quad \text{for } 1 \leq i < k \\
(e_i \vee \neg x_i \vee \neg d_{i+1} \vee \neg e_{i+1}) \quad & \wedge \quad \text{for } 1 \leq i < k \\
(d_k \vee x_k \vee \neg f_1 \vee \ldots \vee \neg f_k) \quad & \wedge \\
(e_k \vee \neg x_k \vee \neg f_1 \vee \ldots \vee \neg f_k) \quad & \wedge \\
(x_i \vee f_i) \wedge (\neg x_i \vee f_i) \quad & \text{for } 1 \leq i \leq k
\end{aligned}$$

(b) Structure of formulas of the KBKF formula family based on Q-resolution for $k > 1$.

Figure 8: Overview of the structure and a specific model of the KBKF formula family.

on Q-resolution, which we will discuss in both the true and false variants.

Hereby, false formulas of the KBKF formula family are constructed as depicted in Figure 8a for $k = 1$ and Figure 8b for $k > 1$. They can also be converted into true formulas by negating the original formula and then applying the Tseitin transformation to convert the resulting formula back to CNF. Negating KBKF formulas and applying Tseitin transformation yields true formulas with the structure described in Figure 9a for $k > 1$.

Solution Count: The solution count for the original KBKF formula is given by the following recursive definition:

$$\#(\Phi_1) = \quad 4$$
$$\#(\Phi_k) = \quad 2 \cdot \#(\Phi_{k-1})^3 \cdot b(k)$$

where $b(1) = 2$ and $b(k) = b(k-1)^4 \cdot 2$. Alternatively we can describe the recursive function $\#(\Phi_k)$ with the closed formula $\#(\Phi_k) = 2^{4^k - 3^k} \cdot 2^{(4^k - 1)/3} = 2^{(4 \cdot 4^k - 3 \cdot 3^k - 1)/3}$.

Theorem 6.1. *For $n \in \mathbb{N}, n > 2$ the model count for the nth true KBKF formula is given by*

$$\#(KBKFTrue(n)) = 2^{(4 \cdot 4^n - 3 \cdot 3^n - 1)/3}. \tag{11}$$

Proof. The proof of the closed form follows by induction on k, and is straightforward. We omit the details for brevity. □

6.1 Evaluation

Key metrics for the true and false KBKF formulas are presented in Table 4, whereas for true formulas we can perform model counting using the model counter d4. Again, we observe an exponential increase in the number of models as the parameter n grows, with a significant rise in runtime for $n > 10$. For formulas with parameters $n = 20$ and $n = 25$, the model counter terminated due to out-of-memory errors.

7 Conclusion

In this paper we have examined several classical formula families from proof complexity, calculating the number of (counter-)models based on the structure of the formulas and combinatorial arguments. We observed that computing the number of models based on their inherent meaning is only possible for some families. In addition, also the Tseitin variables introduced to transform formulas into conjunctive normal form and the prenexing (i.e., the shifting of the quantifiers to the front) might have an impact on the number of solutions. The quantifier structure plays an important role for the computation of the (counter-)model count. Even for formulas with the same inherent meaning and the same propositional matrix the number of (counter-)models may vary significantly. With this work, we provide a huge benchmark set for counting solutions of QBFs which is important for validating the correctness of such solution counters. We also presented key metrics of the presented formulas, as well as experimental results for the model counters d4 and QCounter, empathizing the importance of the encoding and quantifier structure used to generate the formulas. All used formulas and corresponding experimental logs as well as an implementation of an instance generator are available at [15].

In future work we want to extend this collection with benchmarks from real-world encodings. Furthermore, we want to explore other definitions of models, such as *disjoint solutions* [17], which are independent of the quantifier structure of the formulas.

References

[1] Michael Bauland, Elmar Böhler, Nadia Creignou, Steffen Reith, Henning Schnoor, and Heribert Vollmer. 2005. Quantified Constraints: The Complexity of Decision and Counting for Bounded Alternation. *Electron. Colloquium Comput. Complex.* TR05-024 (2005).

[2] Olaf Beyersdorff and Joshua Blinkhorn. 2021. A simple proof of QBF hardness. *Inform. Process. Lett.* 168 (2021), 106093. https://doi.org/10.1016/j.ipl.2021.106093

$$
\begin{aligned}
&\Phi_1 = \forall d_1, e_1 \exists x_1 \forall f_1 \exists l_1 \dots l_5. \\
&(\neg l_1 \vee \neg d_1) \wedge (\neg l_1 \vee \neg x_1) \wedge (\neg l_1 \vee f_1) && \wedge \\
&(\neg l_2 \vee \neg e_1) \wedge (\neg l_2 \vee x_1) \wedge (\neg l_2 \vee f_1) && \wedge \\
&(\neg l_3 \vee \neg x_1) \wedge (\neg l_3 \vee \neg f_1) \wedge (\neg l_4 \vee x_1) \wedge (\neg l_4 \vee \neg f_1) && \wedge \\
&(\neg l_5 \vee d_1) \wedge (\neg l_5 \vee e_1) \wedge (l_1 \vee \dots \vee l_5)
\end{aligned}
$$

(a) Structure of formulas of the true KBKF formula family based on Q-resolution for $k = 1$.

$$
\begin{aligned}
&\Phi_k = \forall d_1, e_1 \exists x_1 \dots \forall d_k e_k \exists x_k \forall f_1 \dots f_k \exists l_1 \dots l_{4k+1}. \\
&(\neg l_i \vee \neg d_i) \wedge (\neg l_i \vee \neg x_i) \wedge (\neg l_i \vee d_{i+1})(\neg l_i \vee e_{i+1}) && \wedge \quad \text{for } 1 \le i < k \\
&(\neg l_{k+i-1} \vee \neg e_i) \wedge (\neg l_{k+i-1} \vee x_i) \wedge (\neg l_{k+i-1} \vee d_{i+1})(\neg l_{k+i-1} \vee e_{i+1}) && \wedge \quad \text{for } 1 \le i < k \\
&(\neg l_{2k-1} \vee \neg d_k) \wedge (\neg l_{2k-1} \vee \neg x_k) \wedge (\neg l_{2k-1} \vee f_1) \dots (\neg l_{2k-1} \vee f_k) && \wedge \\
&(\neg l_{2k} \vee \neg e_k) \wedge (\neg l_{2k} \vee x_k) \wedge (\neg l_{2k} \vee f_1) \dots (\neg l_{2k} \vee f_k) && \wedge \\
&(\neg l_{2k+i} \vee \neg x_i) \wedge (\neg l_{2k+i} \vee \neg f_i) \wedge (\neg l_{3k+i} \vee x_i) \wedge (\neg l_{3k+i} \vee \neg f_i) && \wedge \quad \text{for } 1 \le i \le k \\
&(\neg l_{4k+1} \vee d_1) \wedge (\neg l_{4k+1} \vee e_1) \wedge (l_1 \vee \dots \vee l_{4k+1})
\end{aligned}
$$

(b) Structure of formulas of the true KBKF formula family based on Q-resolution for $k > 1$.

Figure 9: Structure of formulas of the true KBKF formula family based on Q-resolution

n	#Vars	#Clauses	#∃Vars	#∀Vars	#Solutions	Runtime - d4 (in s)
2	8	9	6	2	4096	0.01
5	20	21	15	5	$5.70 \cdot 10^{337}$	0.02
10	40	41	30	10	$6.57 \cdot 10^{403094}$	0.23
15	60	61	45	15	$2.08 \cdot 10^{426651877}$	148.16
20	80	81	60	20	$2.12 \cdot 10^{440265014029}$	TO
25	100	101	75	25	$4.87 \cdot 10^{451651132813455}$	TO
k	$4 \cdot k$	$4 \cdot k + 1$	$3 \cdot k$	k	$2^{(4 \cdot 4^k - 3 \cdot 3^k - 1)/3}$	-

Table 4: Key metrics and runtime for KBKF formulas.

[3] Olaf Beyersdorff, Joshua Blinkhorn, and Luke Hinde. 2017. Size, Cost, and Capacity: A Semantic Technique for Hard Random QBFs. *Logical Methods in Computer Science* Volume 15, Issue 1 (12 2017). https://doi.org/10.23638/LMCS-15(1:13)2019

[4] Olaf Beyersdorff, Leroy Chew, and Mikoláš Janota. 2019. New Resolution-Based QBF Calculi and Their Proof Complexity. *ACM Transactions on Computation Theory* 11 (09 2019), 1–42. https://doi.org/10.1145/3352155

[5] P. Marin E. Giunchiglia and M. Narizzano. 2009. Reasoning with quantified boolean formulas. In *Handbook of Satisfiability*. Frontiers in Artificial Intelligence and Applications, Vol. 185. IOS Press.

[6] Merrick Furst, James B. Saxe, and Michael Sipser. 1981. Parity, circuits, and the polynomial-time hierarchy. In *22nd Annual Symposium on Foundations of Computer Science (sfcs 1981)*. 260–270. https://doi.org/10.1109/SFCS.1981.35

[7] Simone Heisinger and Martina Seidl. 2023. True Crafted Formula Families for Benchmarking Quantified Satisfiability Solvers. In *Intelligent Computer Mathematics*, Catherine Dubois and Manfred Kerber (Eds.). Springer Nature Switzerland, Cham, 291–296.

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

[9] Hans Kleine Büning, Marek Karpinski, and Andreas Flögel. 1995. Resolution for Quantified Boolean Formulas. *Inf. Comput.* 117, 1 (1995), 12–18. https://doi.org/10.1006/INCO.1995.1025

[10] Hans Kleine Büning, Marek Karpinski, and Andreas Flögel. 1995. Resolution for Quantified Boolean Formulas. *Inf. Comput.* 117, 1 (1995), 12–18. https://doi.org/10.1006/INCO.1995.1025

[11] Richard E. Ladner. 1989. Polynomial Space Counting Problems. *SIAM J. Comput.* 18, 6 (1989), 1087–1097.

[12] Jean-Marie Lagniez, Florent Capelli, Andreas Plank, and Martina Seidl. 2024. A Top-Down Tree Model Counter for Quantified Boolean Formulas. In *(unpublished) Proc. of the 33rd. Int. Conf. on International Joint Conference on Artificial Intelligence IJCAI.*

[13] Florian Lonsing. 2019. QBFRelay, QRATPre+, and DepQBF: Incremental Preprocessing Meets Search-Based QBF Solving. *J. Satisf. Boolean Model. Comput.* 11, 1 (2019), 211–220. https://doi.org/10.3233/SAT190122

[14] David A. Plaisted and Steven Greenbaum. 1986. A Structure-preserving Clause Form Translation. *Journal of Symbolic Computation* 2, 3 (1986), 293–304.

[15] Andreas Plank, Manuel Kauers, and Martina Seidl. 2024. *Artifact for "Solution Counts of Some Prominent Quantified Boolean Formulas Families"*. https://doi.org/10.5281/zenodo.14515612

[16] Andreas Plank, Sibylle Möhle, and Martina Seidl. 2023. Enumerative Level-2 Solution Counting for Quantified Boolean Formulas. In *29th Int. Conf. on Principles and Practice of Constraint Programming (CP 2023)*, Vol. 280. 49:1–49:10.

[17] Andreas Plank, Sibylle Möhle, and Martina Seidl. 2023. 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) (LIPIcs, Vol. 280)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 49:1–49:10.

[18] Arijit Shaw, Brendan Juba, and Kuldeep S. Meel. 2024. An Approximate Skolem Function Counter. 38, 8 (Mar. 2024), 8108–8116.

[19] Ankit Shukla, Sibylle Möhle, Manuel Kauers, and Martina Seidl. 2022. OuterCount: A First-Level Solution-Counter for Quantified Boolean Formulas. In *Proc. of the 15th Int. Conf on Intelligent Computer Mathematics (CICM) (LNCS, Vol. 13467)*. Springer, 272–284.

[20] Grigori S Tseitin. 1983. On the complexity of derivation in propositional calculus. *Automation of reasoning: 2: Classical papers on computational logic 1967–1970* (1983), 466–483.

[21] Allen Van Gelder. 2012. Contributions to the Theory of Practical Quantified Boolean Formula Solving. In *Principles and Practice of Constraint Programming*, Michela Milano (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 647–663.